NRL Report 7777

AN/UYK-17(XB-1)(V) Signal Processing Element Microprogramming Support Software

TOMLINSON G. RAUSCHER, JOHN D. ROBERTS, JR.,
AND W. DAVID ELLIOTT

Information Processing Systems Branch
Communications Sciences Division

October 10, 1975

NAVAL RESEARCH LABORATORY
Washington, D.C.

REPORTS IN THIS SERIES

"Microprogrammed Control Unit Programming Reference Manual," J. D. Roberts, Jr., NRL Report 7476, Aug. 15, 1972

"Signal Processing Element Users' Reference Manual," W. R. Smith, and J. P. Ihnat, NRL Report 7488, Sept. 5, 1972

"Signal Processing Element Functional Description, Part 1, Microprogrammed Control Unit, Buffer Store, and Storage Control Unit," J. P. Ihnat, W. R. Smith, J. D. Roberts, Jr., Y. S. Wu, and B. Wald, NRL Report 7490, Sept. 12, 1972

"Signal Processing Element Functional Description, Part 2, Signal Processing Arithmetic Unit," W. R. Smith and H. H. Smith, NRL Memorandum Report 2522, Oct. 1972

"On the Feasibility of Emulating the AN/UYK-7 Computer on the AADC Signal Processing Element," T. G. Rauscher, NRL Memorandum Report 2525, Nov. 1972

"AN/UYK-17 (XB-1)(V) Signal Processing Element Architecture," W. R. Smith, J. P. Ihnat, H. H. Smith, N. M. Head, Jr., E. Freeman, Y. S. Wu, and B. Wald, NRL Report 7704, June 7, 1974

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER NRL Report 7777	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE *(and Subtitle)* AN/UYK-17 (XB-1)(V) SIGNAL PROCESSING ELEMENT MICROPROGRAMMING SUPPORT SOFTWARE		5. TYPE OF REPORT & PERIOD COVERED Interim report on a continuing problem
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Tomlinson G. Rauscher, John D. Roberts, Jr., and W. David Elliott		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Naval Research Laboratory Washington, D.C. 20375		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS NRL Problems B02-06, B02-10, and B02-16 Projects WF21-241-601, YF21-241-019, and ZF11-121-003
11. CONTROLLING OFFICE NAME AND ADDRESS Department of the Navy Naval Air Systems Command and Naval Electronics Systems Command Washington, D.C. 20360		12. REPORT DATE October 10, 1975
		13. NUMBER OF PAGES 124
14. MONITORING AGENCY NAME & ADDRESS *(if different from Controlling Office)*		15. SECURITY CLASS. *(of this report)* Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT *(of this Report)*

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT *(of the abstract entered in Block 20, if different from Report)*

18. SUPPLEMENTARY NOTES

19. KEY WORDS *(Continue on reverse side if necessary and identify by block number)*

AADC	Microprogrammed Control Unit	SPE
All Applications Digital Computer	Microprogramming	
AMIL	Signal processing	
ANIMIL	Signal Processing Arithmetic Unit	
AN/UYK-17 Signal Processing Element	Signal Processing Element	
MCU	SPAU	

20. ABSTRACT *(Continue on reverse side if necessary and identify by block number)*

The AN/UYK-17 Signal Processing Element (SPE) is being developed at the Naval Research Laboratory to provide a high performance processing facility for radar, sonar, and communications systems. The design of the microprogrammable SPE enables realization of efficient, flexible solutions to problems that arise in digital signal processing tasks.

The SPE is intended to be compatible with the Navy All Applications Digital Computer (AADC) now under development, and it is intended to be implemented as part of the AADC system. The SPE can also be used as a stand-alone processor.

(Continued)

20. ABSTRACT

The SPE is a collection of several interconnected functional components. Two of these components, the Microprogrammed Control Unit (MCU) and the Signal Processing Arithmetic Unit (SPAU), are microprogrammable. This report describes the microprogramming support software developed at NRL to facilitate the development and debugging of microprograms for the Model XB-1 MCU and SPAU.

Microprogramming languages, AMIL for the MCU and ANIMIL for the SPAU, have been defined to facilitate microprogram creation. Translators have been developed for each language that convert the source languages to microcode for the MCU and SPAU. Microcode thus produced can be loaded into the control store of the associated machine.

In addition, simulators for the MCU and SPAU have been implemented to aid microprogram debugging and hardware testing. These simulators accept microcode produced by the translators and provide the user with convenient interactive control of the simulated machine. Both the MCU and SPAU simulators use the same user command language to control simulator execution and the display and modification of the simulated processor's registers.

Two ancillary programs have been written to aid the user in writing microprograms. A preprocessor provides translate-time macro facilities for both AMIL and ANIMIL, and a linkage editor allows programs to be written in modules and combined after translation.

CONTENTS

INTRODUCTION 1

AN/UYK-17 Hardware System Overview 1
AN/UYK-17 Support Software System Overview 3

MICROPROGRAMMING THE MCU 3

AMIL 3
Referencing MCU Facilities 5
AMIL Subcommands 6
I/O Instructions 12
Pseudoinstructions 12
The AMIL Translator 14

MICROPROGRAMMING THE SPAU 15

ANIMIL 15
Referencing SPAU Facilities 20
Use of the Arithmetic Section 22
Use of the Address Generator Section 27
Use of the Sequence Unit 29
Use of the IOCU and Buffer Memories 31
Pseudoinstructions 32
ANIMIL Translate-Time Facilities 32
The ANIMIL Translator 33

AN INTERACTIVE SIMULATOR PACKAGE FOR THE MCU AND SPAU 40

Controlling SPE Component Simulators 40
Command Language Overview 40
Command Language Statements 43
Set and Display Commands 44
File Manipulation Commands 48
Output Control Commands 49
Trace Point and Breakpoint Control Commands 50
File Input Command 52
Simulator Control Commands 52
Using the Simulators and Command Language Interpreter 53
Error Messages 53
Summary of Commands 54
Examples 56

20. ABSTRACT

The SPE is a collection of several interconnected functional components. Two of these components, the Microprogrammed Control Unit (MCU) and the Signal Processing Arithmetic Unit (SPAU), are microprogrammable. This report describes the microprogramming support software developed at NRL to facilitate the development and debugging of microprograms for the Model XB-1 MCU and SPAU.

Microprogramming languages, AMIL for the MCU and ANIMIL for the SPAU, have been defined to facilitate microprogram creation. Translators have been developed for each language that convert the source languages to microcode for the MCU and SPAU. Microcode thus produced can be loaded into the control store of the associated machine.

In addition, simulators for the MCU and SPAU have been implemented to aid microprogram debugging and hardware testing. These simulators accept microcode produced by the translators and provide the user with convenient interactive control of the simulated machine. Both the MCU and SPAU simulators use the same user command language to control simulator execution and the display and modification of the simulated processor's registers.

Two ancillary programs have been written to aid the user in writing microprograms. A preprocessor provides translate-time macro facilities for both AMIL and ANIMIL, and a linkage editor allows programs to be written in modules and combined after translation.

AN/UYK-17 (XB-1)(V) SIGNAL PROCESSING ELEMENT MICROPROGRAMMING SUPPORT SOFTWARE

INTRODUCTION

The AN/UYK-17 Signal Processing Element (SPE) is being developed at the Naval Research Laboratory (NRL) to provide a high-performance processing facility for radar, sonar, and communications systems. A prototype version of the SPE, denoted (XB-1), is under construction at NRL. The design of the microprogrammable SPE enables realization of efficient, flexible solutions to problems that arise in digital signal processing tasks.

The SPE is intended to be compatible with and implemented as part of the Navy All Applications Digital Computer (AADC) system under development. The SPE can also be used as a stand-alone processor.

The SPE (see Fig. 1) comprises six modular components that provide the functions of system control, arithmetic processing, storage and its addressing, input/output (I/O) control, and I/O units.

Two of these components, the Microprogrammed Control Unit (MCU) and the Signal Processing Arithmetic Unit (SPAU), are microprogrammable. This report describes the microprogramming support software developed at NRL to facilitate the development and debugging of microprograms for the (XB-1) MCU and (XB-1) SPAU. This report assumes familiarity with an earlier report on AN/UYK-17 SPE architecture [1].

AN/UYK-17 Hardware System Overview

The Microprogrammed Control Unit (MCU) is the SPE system controller. MCU functions include data management, processing scheduling, I/O control, interrupt handling, and applications routine processing. The MCU contains an arithmetic/logic unit, several high-speed local store registers, priority interrupt handling facilities, and I/O logic. The 64-bit horizontal microinstructions, which reside in a 4096 word writable control store, are executed at the system cycle rate of 150 ns.

The SPAU is the SPE system arithmetic processor and operates on arrays of data. The SPAU was designed to perform operations such as Fourier transforms, recursive filtering, and complex multiplication. Parallel multiply and arithmetic/logic units, high-speed

Note: Manuscript submitted May 31, 1974.

20. ABSTRACT

The SPE is a collection of several interconnected functional components. Two of these components, the Microprogrammed Control Unit (MCU) and the Signal Processing Arithmetic Unit (SPAU), are microprogrammable. This report describes the microprogramming support software developed at NRL to facilitate the development and debugging of microprograms for the Model XB-1 MCU and SPAU.

Microprogramming languages, AMIL for the MCU and ANIMIL for the SPAU, have been defined to facilitate microprogram creation. Translators have been developed for each language that convert the source languages to microcode for the MCU and SPAU. Microcode thus produced can be loaded into the control store of the associated machine.

In addition, simulators for the MCU and SPAU have been implemented to aid microprogram debugging and hardware testing. These simulators accept microcode produced by the translators and provide the user with convenient interactive control of the simulated machine. Both the MCU and SPAU simulators use the same user command language to control simulator execution and the display and modification of the simulated processor's registers.

Two ancillary programs have been written to aid the user in writing microprograms. A preprocessor provides translate-time macro facilities for both AMIL and ANIMIL, and a linkage editor allows programs to be written in modules and combined after translation.

20. ABSTRACT

The SPE is a collection of several interconnected functional components. Two of these components, the Microprogrammed Control Unit (MCU) and the Signal Processing Arithmetic Unit (SPAU), are microprogrammable. This report describes the microprogramming support software developed at NRL to facilitate the development and debugging of microprograms for the Model XB-1 MCU and SPAU.

Microprogramming languages, AMIL for the MCU and ANIMIL for the SPAU, have been defined to facilitate microprogram creation. Translators have been developed for each language that convert the source languages to microcode for the MCU and SPAU. Microcode thus produced can be loaded into the control store of the associated machine.

In addition, simulators for the MCU and SPAU have been implemented to aid microprogram debugging and hardware testing. These simulators accept microcode produced by the translators and provide the user with convenient interactive control of the simulated machine. Both the MCU and SPAU simulators use the same user command language to control simulator execution and the display and modification of the simulated processor's registers.

Two ancillary programs have been written to aid the user in writing microprograms. A preprocessor provides translate-time macro facilities for both AMIL and ANIMIL, and a linkage editor allows programs to be written in modules and combined after translation.

20. ABSTRACT

The SPE is a collection of several interconnected functional components. Two of these components, the Microprogrammed Control Unit (MCU) and the Signal Processing Arithmetic Unit (SPAU), are microprogrammable. This report describes the microprogramming support software developed at NRL to facilitate the development and debugging of microprograms for the Model XB-1 MCU and SPAU.

Microprogramming languages, AMIL for the MCU and ANIMIL for the SPAU, have been defined to facilitate microprogram creation. Translators have been developed for each language that convert the source languages to microcode for the MCU and SPAU. Microcode thus produced can be loaded into the control store of the associated machine.

In addition, simulators for the MCU and SPAU have been implemented to aid microprogram debugging and hardware testing. These simulators accept microcode produced by the translators and provide the user with convenient interactive control of the simulated machine. Both the MCU and SPAU simulators use the same user command language to control simulator execution and the display and modification of the simulated processor's registers.

Two ancillary programs have been written to aid the user in writing microprograms. A preprocessor provides translate-time macro facilities for both AMIL and ANIMIL, and a linkage editor allows programs to be written in modules and combined after translation.

20. ABSTRACT

The SPE is a collection of several interconnected functional components. Two of these components, the Microprogrammed Control Unit (MCU) and the Signal Processing Arithmetic Unit (SPAU), are microprogrammable. This report describes the microprogramming support software developed at NRL to facilitate the development and debugging of microprograms for the Model XB-1 MCU and SPAU.

Microprogramming languages, AMIL for the MCU and ANIMIL for the SPAU, have been defined to facilitate microprogram creation. Translators have been developed for each language that convert the source languages to microcode for the MCU and SPAU. Microcode thus produced can be loaded into the control store of the associated machine.

In addition, simulators for the MCU and SPAU have been implemented to aid microprogram debugging and hardware testing. These simulators accept microcode produced by the translators and provide the user with convenient interactive control of the simulated machine. Both the MCU and SPAU simulators use the same user command language to control simulator execution and the display and modification of the simulated processor's registers.

Two ancillary programs have been written to aid the user in writing microprograms. A pre-processor provides translate-time macro facilities for both AMIL and ANIMIL, and a linkage editor allows programs to be written in modules and combined after translation.

20. ABSTRACT

The SPE is a collection of several interconnected functional components. Two of these components, the Microprogrammed Control Unit (MCU) and the Signal Processing Arithmetic Unit (SPAU), are microprogrammable. This report describes the microprogramming support software developed at NRL to facilitate the development and debugging of microprograms for the Model XB-1 MCU and SPAU.

Microprogramming languages, AMIL for the MCU and ANIMIL for the SPAU, have been defined to facilitate microprogram creation. Translators have been developed for each language that convert the source languages to microcode for the MCU and SPAU. Microcode thus produced can be loaded into the control store of the associated machine.

In addition, simulators for the MCU and SPAU have been implemented to aid microprogram debugging and hardware testing. These simulators accept microcode produced by the translators and provide the user with convenient interactive control of the simulated machine. Both the MCU and SPAU simulators use the same user command language to control simulator execution and the display and modification of the simulated processor's registers.

Two ancillary programs have been written to aid the user in writing microprograms. A preprocessor provides translate-time macro facilities for both AMIL and ANIMIL, and a linkage editor allows programs to be written in modules and combined after translation.

20. ABSTRACT

The SPE is a collection of several interconnected functional components. Two of these components, the Microprogrammed Control Unit (MCU) and the Signal Processing Arithmetic Unit (SPAU), are microprogrammable. This report describes the microprogramming support software developed at NRL to facilitate the development and debugging of microprograms for the Model XB-1 MCU and SPAU.

Microprogramming languages, AMIL for the MCU and ANIMIL for the SPAU, have been defined to facilitate microprogram creation. Translators have been developed for each language that convert the source languages to microcode for the MCU and SPAU. Microcode thus produced can be loaded into the control store of the associated machine.

In addition, simulators for the MCU and SPAU have been implemented to aid microprogram debugging and hardware testing. These simulators accept microcode produced by the translators and provide the user with convenient interactive control of the simulated machine. Both the MCU and SPAU simulators use the same user command language to control simulator execution and the display and modification of the simulated processor's registers.

Two ancillary programs have been written to aid the user in writing microprograms. A preprocessor provides translate-time macro facilities for both AMIL and ANIMIL, and a linkage editor allows programs to be written in modules and combined after translation.

are the following (see field 12): +1, –1, COMP, LEFT, RIGHT, and CIRC. A blank must precede each of the last four unary operations (which indicate one's complement and left, right, and circular shift by the value of the SAR register). The binary operators (indicated by binary-op) are the following (see field 12): +, –, OR, AND, XOR, and EQV. The last four operators are logical operators and must be surrounded by blanks in the subcommand. All the binary operators except "–" are commutative, and the order of the left and right operands is immaterial. Note that when binary operators are used, the only valid combinations of left and right inputs are those listed in field 11. Note in fields 4 and 5 of the control word that the BARA and BARB registers receive their inputs from the ALU/shifter and not from the Z register. Hence the subcommands

```
BARA = 0
BARB = 0
```

may be performed by doing a no-operation in the ALU and assigning the result, zero, to BARA or BARB. In this case, the Z register remains unchanged and other registers may be assigned its value with a subcommand of the form

```
destination = Z
```

Note that adder destinations may not conflict with other uses of a register. For example, local store A (see field 6) can be set from channel A, from channel B, or from the Z register. Some legal examples of ALU operations are the following:

```
LSB(3) = LSA(4) + LSB(4) $
LSA(1), SAR = BARA + 10 $ (note—two destinations)
ACSAR = LSB(12)$(+ 0 need not be included)
LSA(1) = LSA(1) OR LSB(7) $
Z = LSA(9) + 65535 $
LSB(5) = LSB(5) COMP $ (LSB(5) is one's complemented)
BARB = LSB(10) + COMP1 36 $ (36 is added to LSB(10))
LSA(3) = LSA(3) LEFT $ (LSA(3) is Left Shifted SAR bits)
LSA(1), LSB(2), BARA, BARB, ACSAR = LSA(5) EQV LSB(2) $
*many destinations
SAR, LSA(1) = BARA + 10 $
```

Some illegal examples of ALU operations follow:

```
LSA(7) = CTR + LSA(5) $ Illegal Left/Right combination
BARA = LSB(5) - LSA(3) $ only Left-Right
ACSAR = SAR + CTR $ Illegal Left/Right combination
LSB(3) = 537 RIGHT $ Lit is not a Left input
CTR = COMP LSA(5) $ Should be LSA(5) COMP
```

Address Adjust—In the latter part of each MCU cycle, the buffer address registers, BARA and BARB, may be changed to properly address the buffers in the next cycle

20. ABSTRACT

The SPE is a collection of several interconnected functional components. Two of these components, the Microprogrammed Control Unit (MCU) and the Signal Processing Arithmetic Unit (SPAU), are microprogrammable. This report describes the microprogramming support software developed at NRL to facilitate the development and debugging of microprograms for the Model XB-1 MCU and SPAU.

Microprogramming languages, AMIL for the MCU and ANIMIL for the SPAU, have been defined to facilitate microprogram creation. Translators have been developed for each language that convert the source languages to microcode for the MCU and SPAU. Microcode thus produced can be loaded into the control store of the associated machine.

In addition, simulators for the MCU and SPAU have been implemented to aid microprogram debugging and hardware testing. These simulators accept microcode produced by the translators and provide the user with convenient interactive control of the simulated machine. Both the MCU and SPAU simulators use the same user command language to control simulator execution and the display and modification of the simulated processor's registers.

Two ancillary programs have been written to aid the user in writing microprograms. A preprocessor provides translate-time macro facilities for both AMIL and ANIMIL, and a linkage editor allows programs to be written in modules and combined after translation.

20. ABSTRACT

The SPE is a collection of several interconnected functional components. Two of these components, the Microprogrammed Control Unit (MCU) and the Signal Processing Arithmetic Unit (SPAU), are microprogrammable. This report describes the microprogramming support software developed at NRL to facilitate the development and debugging of microprograms for the Model XB-1 MCU and SPAU.

Microprogramming languages, AMIL for the MCU and ANIMIL for the SPAU, have been defined to facilitate microprogram creation. Translators have been developed for each language that convert the source languages to microcode for the MCU and SPAU. Microcode thus produced can be loaded into the control store of the associated machine.

In addition, simulators for the MCU and SPAU have been implemented to aid microprogram debugging and hardware testing. These simulators accept microcode produced by the translators and provide the user with convenient interactive control of the simulated machine. Both the MCU and SPAU simulators use the same user command language to control simulator execution and the display and modification of the simulated processor's registers.

Two ancillary programs have been written to aid the user in writing microprograms. A preprocessor provides translate-time macro facilities for both AMIL and ANIMIL, and a linkage editor allows programs to be written in modules and combined after translation.

```
IF MOST THEN JUMP TO ERROR $
IF NOT ZERO THEN CALL MULT $
IF NOT ADROV THEN SKIP $
```

Table 2
MCU Conditions*

Mnemonic	Meaning
MOST	The most significant bit of the result of the last ALU operation is set
LEAST	The least significant bit of the result of the last ALU operation is set
ADROV	The most significant bit of the ALU operands are the same yet different from the most significant bit of result
CTRZERO	Counter is zero (after the test the counter is decremented)
ZERO	The result of the last ALU operation was zero
FSU	The defined FSU field (at the beginning of the cycle) is zero
CARRY	There was a carry out of the most significant bit during the last ALU operation

*Conditions must be surrounded by blanks.

Interrupt Control Adjust—The MCU programmer can set or reset the two bits of the interrupt status register (see field 10) that designate an inhibit interrupt flag and a software interrupt flag. The subcommand

ISR = literal

indicates these actions. The low-order four bits of the literal are, from most to least significant, "set software interrupt flag," "clear software interrupt flag," "set inhibit interrupt flag," and "clear inhibit interrupt flag." A 1 in the appropriate bit position specifies the listed action. The set and clear operations for each flag are mutually exclusive. The following subcommand, for example, sets the software interrupt flag and clears the inhibit interrupt flag:

ISR = 9

Field Select Control Adjust—Data from the FSDR, indicated by the key word FSU, may be referenced as an ALU input, as a source for BARB, or as a condition. Because the FSDR is 32 bits long, there is a Field Select Control Register (FSCR) that indicates which bits are to be selected from the FSDR. The FSCR is a 10-bit register that indicates the rightmost bit position in the FSDR from which to select bits and the number of bits to select. See diagram below.

20. ABSTRACT

The SPE is a collection of several interconnected functional components. Two of these components, the Microprogrammed Control Unit (MCU) and the Signal Processing Arithmetic Unit (SPAU), are microprogrammable. This report describes the microprogramming support software developed at NRL to facilitate the development and debugging of microprograms for the Model XB-1 MCU and SPAU.

Microprogramming languages, AMIL for the MCU and ANIMIL for the SPAU, have been defined to facilitate microprogram creation. Translators have been developed for each language that convert the source languages to microcode for the MCU and SPAU. Microcode thus produced can be loaded into the control store of the associated machine.

In addition, simulators for the MCU and SPAU have been implemented to aid microprogram debugging and hardware testing. These simulators accept microcode produced by the translators and provide the user with convenient interactive control of the simulated machine. Both the MCU and SPAU simulators use the same user command language to control simulator execution and the display and modification of the simulated processor's registers.

Two ancillary programs have been written to aid the user in writing microprograms. A preprocessor provides translate-time macro facilities for both AMIL and ANIMIL, and a linkage editor allows programs to be written in modules and combined after translation.

The equivalence pseudoinstruction permits mnemonic referencing of constants and certain registers. The general form of the command is

EQU mnemonic = equ-factor

One or more blanks must follow the key word EQU. The mnemonic has the same form as a label. The equ-factor may be one of the following:

1. An integer,

2. A hexadecimal number indicated by

HEX hexadecimal-integer

(a blank must follow the key word HEX),

3. A complemented integer indicated by

COMP1 integer

or

COMP2 integer

depending on whether one's or two's complement is desired (one or more blanks must follow the key words COMP1 and COMP2),

4. A buffer address indicated by

$(\underline{\text{integer}_1}, \underline{\text{integer}_2})$

where $\underline{\text{integer}_1}$ represents the buffer number and $\underline{\text{integer}_2}$ represents the address of a 16-bit word, or

5. One of the registers BARA, BARB, LSA(subscript), or LSB(subscript).

In the first four cases, the number is converted to a 16-bit integer that generally represents a control store literal. When a mnemonic is later encountered in an AMIL program, the value assigned to the mnemonic is substituted for it. Mnemonics for BARA and BARB may not be used in memory operations. Once a mnemonic has been assigned, its value cannot be changed. Some examples of use of the EQU instruction are as follows:

```
EQU ADDR1   = (3,512) $
EQU TWO10   = 1024 $
EQU ABLE    = BARA $
EQU BAKER   = LSA(2) $
EQU CHARLIE = LSB(12) $

BARA = ADDR1 $
ABLE = BAKER + CHARLIE $
CTR  = TWO10 $
```

20. ABSTRACT

The SPE is a collection of several interconnected functional components. Two of these components, the Microprogrammed Control Unit (MCU) and the Signal Processing Arithmetic Unit (SPAU), are microprogrammable. This report describes the microprogramming support software developed at NRL to facilitate the development and debugging of microprograms for the Model XB-1 MCU and SPAU.

Microprogramming languages, AMIL for the MCU and ANIMIL for the SPAU, have been defined to facilitate microprogram creation. Translators have been developed for each language that convert the source languages to microcode for the MCU and SPAU. Microcode thus produced can be loaded into the control store of the associated machine.

In addition, simulators for the MCU and SPAU have been implemented to aid microprogram debugging and hardware testing. These simulators accept microcode produced by the translators and provide the user with convenient interactive control of the simulated machine. Both the MCU and SPAU simulators use the same user command language to control simulator execution and the display and modification of the simulated processor's registers.

Two ancillary programs have been written to aid the user in writing microprograms. A preprocessor provides translate-time macro facilities for both AMIL and ANIMIL, and a linkage editor allows programs to be written in modules and combined after translation.

20. ABSTRACT

The SPE is a collection of several interconnected functional components. Two of these components, the Microprogrammed Control Unit (MCU) and the Signal Processing Arithmetic Unit (SPAU), are microprogrammable. This report describes the microprogramming support software developed at NRL to facilitate the development and debugging of microprograms for the Model XB-1 MCU and SPAU.

Microprogramming languages, AMIL for the MCU and ANIMIL for the SPAU, have been defined to facilitate microprogram creation. Translators have been developed for each language that convert the source languages to microcode for the MCU and SPAU. Microcode thus produced can be loaded into the control store of the associated machine.

In addition, simulators for the MCU and SPAU have been implemented to aid microprogram debugging and hardware testing. These simulators accept microcode produced by the translators and provide the user with convenient interactive control of the simulated machine. Both the MCU and SPAU simulators use the same user command language to control simulator execution and the display and modification of the simulated processor's registers.

Two ancillary programs have been written to aid the user in writing microprograms. A pre-processor provides translate-time macro facilities for both AMIL and ANIMIL, and a linkage editor allows programs to be written in modules and combined after translation.

20. ABSTRACT

The SPE is a collection of several interconnected functional components. Two of these components, the Microprogrammed Control Unit (MCU) and the Signal Processing Arithmetic Unit (SPAU), are microprogrammable. This report describes the microprogramming support software developed at NRL to facilitate the development and debugging of microprograms for the Model XB-1 MCU and SPAU.

Microprogramming languages, AMIL for the MCU and ANIMIL for the SPAU, have been defined to facilitate microprogram creation. Translators have been developed for each language that convert the source languages to microcode for the MCU and SPAU. Microcode thus produced can be loaded into the control store of the associated machine.

In addition, simulators for the MCU and SPAU have been implemented to aid microprogram debugging and hardware testing. These simulators accept microcode produced by the translators and provide the user with convenient interactive control of the simulated machine. Both the MCU and SPAU simulators use the same user command language to control simulator execution and the display and modification of the simulated processor's registers.

Two ancillary programs have been written to aid the user in writing microprograms. A preprocessor provides translate-time macro facilities for both AMIL and ANIMIL, and a linkage editor allows programs to be written in modules and combined after translation.

20. ABSTRACT

The SPE is a collection of several interconnected functional components. Two of these components, the Microprogrammed Control Unit (MCU) and the Signal Processing Arithmetic Unit (SPAU), are microprogrammable. This report describes the microprogramming support software developed at NRL to facilitate the development and debugging of microprograms for the Model XB-1 MCU and SPAU.

Microprogramming languages, AMIL for the MCU and ANIMIL for the SPAU, have been defined to facilitate microprogram creation. Translators have been developed for each language that convert the source languages to microcode for the MCU and SPAU. Microcode thus produced can be loaded into the control store of the associated machine.

In addition, simulators for the MCU and SPAU have been implemented to aid microprogram debugging and hardware testing. These simulators accept microcode produced by the translators and provide the user with convenient interactive control of the simulated machine. Both the MCU and SPAU simulators use the same user command language to control simulator execution and the display and modification of the simulated processor's registers.

Two ancillary programs have been written to aid the user in writing microprograms. A preprocessor provides translate-time macro facilities for both AMIL and ANIMIL, and a linkage editor allows programs to be written in modules and combined after translation.

20. ABSTRACT

The SPE is a collection of several interconnected functional components. Two of these components, the Microprogrammed Control Unit (MCU) and the Signal Processing Arithmetic Unit (SPAU), are microprogrammable. This report describes the microprogramming support software developed at NRL to facilitate the development and debugging of microprograms for the Model XB-1 MCU and SPAU.

Microprogramming languages, AMIL for the MCU and ANIMIL for the SPAU, have been defined to facilitate microprogram creation. Translators have been developed for each language that convert the source languages to microcode for the MCU and SPAU. Microcode thus produced can be loaded into the control store of the associated machine.

In addition, simulators for the MCU and SPAU have been implemented to aid microprogram debugging and hardware testing. These simulators accept microcode produced by the translators and provide the user with convenient interactive control of the simulated machine. Both the MCU and SPAU simulators use the same user command language to control simulator execution and the display and modification of the simulated processor's registers.

Two ancillary programs have been written to aid the user in writing microprograms. A preprocessor provides translate-time macro facilities for both AMIL and ANIMIL, and a linkage editor allows programs to be written in modules and combined after translation.

```
         SAME L.S. USED TWICE AS SOURCE OR DESTINATION
        ILLEGAL LOCAL STORE SUBSCRIPT
        MISSING ) AFTER LOCAL STORE SUBSCRIPT
        LOCAL STORE SUBSCRIPT *13* IS TOO LARGE
        PANEL STARTS WITH ILLEGAL CHARACTER
        MISSING ( AFTER INPUT OR OUTPUT
        BUFFER READA OR READB SPECIFIED TWICE
        ILLEGAL DESTINATION FOR BUFFER READ
        ILLEGAL COMBINATION OF BUFFER OUTPUTS
        ILLEGAL ADDRESS SELECT FOR OUTPUT
        MISSING , AFTER SOURCE FOR OUTPUT OR INPUT
        ILLEGAL MODIFIER FOLLOWING INC
        BARA USED TWICE AS A DESTINATION
        BARB USED TWICE AS A DESTINATION
        DUPLICATE AUXILLARY TRANSFER
        MISSING = IN AUXILLARY TRANSFER
        ILLEGAL SOURCE FOR AUXILLARY TRANSFER
        LABEL LONGER THAN NINE CHARACTERS
        UNKNOWN ERROR AT (OR AROUND) PHRASE
        MISSING = IN ADDER OPERATION
        ILLEGAL OPERAND IN ADDER OPERATION
        ILLEGAL INPUT OPERAND COMBINATION FOR ADDER
        UNKNOWN ADDER OPERATOR
        ILLEGAL CONDITION
        ILLEGAL FSCR WIDTH SPECIFICATION
        ILLEGAL FSCR STARTING BIT NUMBER SPECIFICATION
        ILLEGAL FSCR WIDTH AND STARTING BIT NUM. COMBO
        FSCR CANNOT SHARE THE CURRENT LIT FIELD
        ILLEGAL FSCR ALTERATION
        ILLEGAL SOURCE FOR FSDR LOAD
        ILLEGAL SOURCE FOR OUTPUT
        ILLEGAL SOURCE FOR INPUT I/O INSTR.
        ILLEGAL INTERRUPT STATUS REGISTER CHANGE INSTR.
        ILLEGAL FORMAT FOR JUMP OR CALL SUCCESSOR
        DUPLICATE USE OF FSU OR FSDR
        ILLEGAL ORIGIN STATEMENT
        ILLEGAL I/O STATEMENT
        ILLEGAL EQUIVALENCE STATEMENT
        BARA USED FOR READ & WRITE IN SAME CYCLE
        BARB USED FOR READ & WRITE IN SAME CYCLE
        SYMBOLIC LENGTH TOO LONG (> 10)
        OVER 40 LABELS (SYMBOLICS) START WITH
WARNING EOF ENCOUNTERED AFTER LINE #
     LABEL *A9* HAS ALREADY BEEN USED
        EXTERNAL LIST OVERFLOW - MORE THAN 200 ENTRIES
        ENTRY POIN TABLE OVERFLOW - OVER 100 E.P.'S
        DUPLICATE FSCR ALTERATION
        DUPLICATE ISR ALTERATION
        DUPLICATE ADDER OPERATION
         INTEGER MORE THAN 6 DIGITS LONG
        TWO DIFFERENT LITERALS SPECIFIED

        INTEGER > 65535 OR < -32768
         ILLEGAL CHARACTER INSIDE INTEGER
UNABLE TO RESOLVE THE FOLLOWING LABEL(S)
        DUPLICATE LITERALS AT LINE NUMBER
        MISSING ) TO CLOSE INPUT OR OUTPUT STAT
          GREATER THAN 9 IN SUBROUTINE BUILD
         UNABLE TO LOCATE
          ILLEGAL PLACEMENT OF * OR *
       SOLUTION - LINE ENDED WITHOUT CLOSING
```

Fig. 5—Sample error messages for the AMIL

20. ABSTRACT

The SPE is a collection of several interconnected functional components. Two of these components, the Microprogrammed Control Unit (MCU) and the Signal Processing Arithmetic Unit (SPAU), are microprogrammable. This report describes the microprogramming support software developed at NRL to facilitate the development and debugging of microprograms for the Model XB-1 MCU and SPAU.

Microprogramming languages, AMIL for the MCU and ANIMIL for the SPAU, have been defined to facilitate microprogram creation. Translators have been developed for each language that convert the source languages to microcode for the MCU and SPAU. Microcode thus produced can be loaded into the control store of the associated machine.

In addition, simulators for the MCU and SPAU have been implemented to aid microprogram debugging and hardware testing. These simulators accept microcode produced by the translators and provide the user with convenient interactive control of the simulated machine. Both the MCU and SPAU simulators use the same user command language to control simulator execution and the display and modification of the simulated processor's registers.

Two ancillary programs have been written to aid the user in writing microprograms. A preprocessor provides translate-time macro facilities for both AMIL and ANIMIL, and a linkage editor allows programs to be written in modules and combined after translation.

```
 LABEL# LABEL# LABEL# $

LABEL#LABEL#LABEL#LABEL#LABEL#LABEL#SUBCOMMAND;SUBCOMMAND;SUBCOMMAND;SUB
      COMMAND;SUBCOMMAND;SUBCOMMAND;SUBCOMMANDS$

SU
  BCOM
      MANDS

 LAB EL#  SUBCO " HERE'S A COMMENT IN THE MIDDLE OF A SUBCOMMAND " MMAND;
      SUBCOMMAND;  SUBCOMMAND;  S U B C O M M A N D;  $
```

Fig. 6—Format for a sample ANIMIL microprogram (Continued)

The ANIMIL translator translates an ANIMIL microprogram into an object program. Each microinstruction comprises 63 control fields. The names of these fields and their meanings appear in Appendix B, which contains a summary chart that can be folded out for ease of reference. The following subsections discuss referencing SPAU facilities; subcommands that specify use of arithmetic, address generator, sequence unit, and I/O control unit (IOCU) sections; pseudoinstructions; translator use; sample microprograms; and error messages. A companion document describes the formal syntax and semantics of ANIMIL [3].

Referencing SPAU Facilities

The SPAU registers are listed below.

Single Registers

ACSAR	INCB	R2
BARA	INCR	R3
BARB	P1	R4
CSAR	P2	R7
CTRI	P3	Z1
CTRJ	P4	Z2
CTRK	RAR	Z3
INCA	R1	Z4

Array Registers

W	
X	
X1	left half of the X array
X2	right half of the X array
Y	
Y1	left half of the Y array
Y2	right half of the Y array

Memory Registers

BUFA
BUFB
ROM

Note that the ACSAR and CSAR registers are 12 bits long. All other single and array registers are 16 bits long. Each buffer memory word is 32 bits long. The first 1025 ROM words are 32 bits long; additional ROM words are 64 bits long.

20. ABSTRACT

The SPE is a collection of several interconnected functional components. Two of these components, the Microprogrammed Control Unit (MCU) and the Signal Processing Arithmetic Unit (SPAU), are microprogrammable. This report describes the microprogramming support software developed at NRL to facilitate the development and debugging of microprograms for the Model XB-1 MCU and SPAU.

Microprogramming languages, AMIL for the MCU and ANIMIL for the SPAU, have been defined to facilitate microprogram creation. Translators have been developed for each language that convert the source languages to microcode for the MCU and SPAU. Microcode thus produced can be loaded into the control store of the associated machine.

In addition, simulators for the MCU and SPAU have been implemented to aid microprogram debugging and hardware testing. These simulators accept microcode produced by the translators and provide the user with convenient interactive control of the simulated machine. Both the MCU and SPAU simulators use the same user command language to control simulator execution and the display and modification of the simulated processor's registers.

Two ancillary programs have been written to aid the user in writing microprograms. A preprocessor provides translate-time macro facilities for both AMIL and ANIMIL, and a linkage editor allows programs to be written in modules and combined after translation.

$.5_{10} \times .25_{10}$

$= 0.100\ 0000\ 0000\ 0000_2 \times 0.010\ 0000\ 0000\ 0000_2$

$= 00.00\ 1000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000_2$

$= 0.00\ 1000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000_2$ After the leftmost bit is dropped

$= 0.001\ 0000\ 0000\ 0000_2$ After rounding and taking 16 bits, starting with bit 30

$= .125_{10}$

The subcommand

P = BIT29;

indicates that the 16 bits starting with bit 29 after rounding will be loaded into the P registers. This choice represents the product of a number whose assumed radix point follows the bit after the sign bit and a fractional number; for example,

$1.5_{10} \times .5_{10}$

$= 01.10\ 0000\ 0000\ 0000_2 \times 0.100\ 0000\ 0000\ 0000_2$

$= 000.1\ 1000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000_2$

$= 00.1\ 1000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000_2$ After the leftmost bit is dropped

$= 0.110\ 0000\ 0000\ 0000_2$ After rounding and taking 16 bits, starting with bit 29

$= .75_{10}$

The subcommand

P = BIT27;

indicates that 16 bits starting with bit 27 after rounding will be loaded into the P registers. This choice represents the product of a number whose assumed radix point follows the third bit after the sign bit and a fractional number; for example,

$6.5_{10} \times .125_{10}$

$= 0110.\ 1000\ 0000\ 0000 \times 0.001\ 0000\ 0000\ 0000_2$

$= 0000\ 0.110\ 1000\ 0000\ 0000\ 0000\ 0000\ 0000_2$

$= 0000.110\ 1000\ 0000\ 0000\ 0000\ 0000\ 0000_2$ After the leftmost bit is dropped

20. ABSTRACT

The SPE is a collection of several interconnected functional components. Two of these components, the Microprogrammed Control Unit (MCU) and the Signal Processing Arithmetic Unit (SPAU), are microprogrammable. This report describes the microprogramming support software developed at NRL to facilitate the development and debugging of microprograms for the Model XB-1 MCU and SPAU.

Microprogramming languages, AMIL for the MCU and ANIMIL for the SPAU, have been defined to facilitate microprogram creation. Translators have been developed for each language that convert the source languages to microcode for the MCU and SPAU. Microcode thus produced can be loaded into the control store of the associated machine.

In addition, simulators for the MCU and SPAU have been implemented to aid microprogram debugging and hardware testing. These simulators accept microcode produced by the translators and provide the user with convenient interactive control of the simulated machine. Both the MCU and SPAU simulators use the same user command language to control simulator execution and the display and modification of the simulated processor's registers.

Two ancillary programs have been written to aid the user in writing microprograms. A preprocessor provides translate-time macro facilities for both AMIL and ANIMIL, and a linkage editor allows programs to be written in modules and combined after translation.

20. ABSTRACT

The SPE is a collection of several interconnected functional components. Two of these components, the Microprogrammed Control Unit (MCU) and the Signal Processing Arithmetic Unit (SPAU), are microprogrammable. This report describes the microprogramming support software developed at NRL to facilitate the development and debugging of microprograms for the Model XB-1 MCU and SPAU.

Microprogramming languages, AMIL for the MCU and ANIMIL for the SPAU, have been defined to facilitate microprogram creation. Translators have been developed for each language that convert the source languages to microcode for the MCU and SPAU. Microcode thus produced can be loaded into the control store of the associated machine.

In addition, simulators for the MCU and SPAU have been implemented to aid microprogram debugging and hardware testing. These simulators accept microcode produced by the translators and provide the user with convenient interactive control of the simulated machine. Both the MCU and SPAU simulators use the same user command language to control simulator execution and the display and modification of the simulated processor's registers.

Two ancillary programs have been written to aid the user in writing microprograms. A preprocessor provides translate-time macro facilities for both AMIL and ANIMIL, and a linkage editor allows programs to be written in modules and combined after translation.

20. ABSTRACT

The SPE is a collection of several interconnected functional components. Two of these components, the Microprogrammed Control Unit (MCU) and the Signal Processing Arithmetic Unit (SPAU), are microprogrammable. This report describes the microprogramming support software developed at NRL to facilitate the development and debugging of microprograms for the Model XB-1 MCU and SPAU.

Microprogramming languages, AMIL for the MCU and ANIMIL for the SPAU, have been defined to facilitate microprogram creation. Translators have been developed for each language that convert the source languages to microcode for the MCU and SPAU. Microcode thus produced can be loaded into the control store of the associated machine.

In addition, simulators for the MCU and SPAU have been implemented to aid microprogram debugging and hardware testing. These simulators accept microcode produced by the translators and provide the user with convenient interactive control of the simulated machine. Both the MCU and SPAU simulators use the same user command language to control simulator execution and the display and modification of the simulated processor's registers.

Two ancillary programs have been written to aid the user in writing microprograms. A preprocessor provides translate-time macro facilities for both AMIL and ANIMIL, and a linkage editor allows programs to be written in modules and combined after translation.

20. ABSTRACT

The SPE is a collection of several interconnected functional components. Two of these components, the Microprogrammed Control Unit (MCU) and the Signal Processing Arithmetic Unit (SPAU), are microprogrammable. This report describes the microprogramming support software developed at NRL to facilitate the development and debugging of microprograms for the Model XB-1 MCU and SPAU.

Microprogramming languages, AMIL for the MCU and ANIMIL for the SPAU, have been defined to facilitate microprogram creation. Translators have been developed for each language that convert the source languages to microcode for the MCU and SPAU. Microcode thus produced can be loaded into the control store of the associated machine.

In addition, simulators for the MCU and SPAU have been implemented to aid microprogram debugging and hardware testing. These simulators accept microcode produced by the translators and provide the user with convenient interactive control of the simulated machine. Both the MCU and SPAU simulators use the same user command language to control simulator execution and the display and modification of the simulated processor's registers.

Two ancillary programs have been written to aid the user in writing microprograms. A preprocessor provides translate-time macro facilities for both AMIL and ANIMIL, and a linkage editor allows programs to be written in modules and combined after translation.

20. ABSTRACT

The SPE is a collection of several interconnected functional components. Two of these components, the Microprogrammed Control Unit (MCU) and the Signal Processing Arithmetic Unit (SPAU), are microprogrammable. This report describes the microprogramming support software developed at NRL to facilitate the development and debugging of microprograms for the Model XB-1 MCU and SPAU.

Microprogramming languages, AMIL for the MCU and ANIMIL for the SPAU, have been defined to facilitate microprogram creation. Translators have been developed for each language that convert the source languages to microcode for the MCU and SPAU. Microcode thus produced can be loaded into the control store of the associated machine.

In addition, simulators for the MCU and SPAU have been implemented to aid microprogram debugging and hardware testing. These simulators accept microcode produced by the translators and provide the user with convenient interactive control of the simulated machine. Both the MCU and SPAU simulators use the same user command language to control simulator execution and the display and modification of the simulated processor's registers.

Two ancillary programs have been written to aid the user in writing microprograms. A preprocessor provides translate-time macro facilities for both AMIL and ANIMIL, and a linkage editor allows programs to be written in modules and combined after translation.

```
CTRI = W(10);
CTRK = LSH W(23);
CTRJ = LIT; ... LIT = 1024;
```

Counters may alternatively be decremented by 1 so that they can be tested (see field 16). The DEC command followed by the letter of the counter to be decremented specifies this action. The three possible subcommands to decrement counters are

```
DECI;
DECJ;
DECK;
```

Use of the Sequence Unit

There are several methods for choosing the control store address (see field 30) of the next microinstruction. Some of these commands use labels, which are strings of 1 to 10 characters, the first of which must be alphabetic and the remainder of which must be alphanumeric. A label names a microinstruction and prefixes it as follows:

label # microinstruction $

A microinstruction may have multiple labels. Microinstruction labels are converted to numeric addresses of control store words wherein the microinstruction will reside. References to other microinstructions are made via labels that are converted to the appropriate control store address and stored in the literal field of the control word. Table 3 contains the subcommand forms that correspond to the alternative methods for selecting the next control store address (see field 30).

Table 3
Alternative Methods of Control Store Address Selection

Subcommand Form	Meaning	Register Action
STEP;	Add one to current address (default)	CSAR←CSAR+1
SKIP;	Add two to current address	CSAR←CSAR+2
SAVE;	Set ACSAR to one plus contents of CSAR and step	ACSAR←CSAR+1,CSAR←CSAR+1
CALL label;	Set ACSAR to one plus contents of CSAR and jump to literal	ACSAR←CSAR+1,CSAR←literal
GO TO label;	Jump to literal	CSAR←literal
GO TO ACSAR;	Jump to contents of ACSAR	CSAR←ACSAR
GO TO W(subscript)	Jump to contents of W(subscript)	CSAR←W(subscript)
GO TO R7;	Jump to contents of R7	CSAR←R7

20. ABSTRACT

The SPE is a collection of several interconnected functional components. Two of these components, the Microprogrammed Control Unit (MCU) and the Signal Processing Arithmetic Unit (SPAU), are microprogrammable. This report describes the microprogramming support software developed at NRL to facilitate the development and debugging of microprograms for the Model XB-1 MCU and SPAU.

Microprogramming languages, AMIL for the MCU and ANIMIL for the SPAU, have been defined to facilitate microprogram creation. Translators have been developed for each language that convert the source languages to microcode for the MCU and SPAU. Microcode thus produced can be loaded into the control store of the associated machine.

In addition, simulators for the MCU and SPAU have been implemented to aid microprogram debugging and hardware testing. These simulators accept microcode produced by the translators and provide the user with convenient interactive control of the simulated machine. Both the MCU and SPAU simulators use the same user command language to control simulator execution and the display and modification of the simulated processor's registers.

Two ancillary programs have been written to aid the user in writing microprograms. A preprocessor provides translate-time macro facilities for both AMIL and ANIMIL, and a linkage editor allows programs to be written in modules and combined after translation.

Use of the IOCU and Buffer Memories

Data computed by a SPAU program may be written in 32-bit words into the buffer memories (see fields 19 and 20) with a subcommand of the form

buffer = buffer-source;

where buffer must be BUFA or BUFB and is addressed by BARA or BARB, respectively. A buffer may not be used as a destination of information in the same microinstruction in which it is a source. Buffer-source may be R1R2, R1R3, R2R4, R3R4, X(subscript), or Y(subscript). The sources may thus be considered pairs of 16-bit half-words. The contents of the first are written into the 16 MSBs of the addressed buffer memory word; the contents of the second are written into the 16 LSBs. Typical examples follow:

```
BUFA = R1R3;
BUFA = X(7);
BUFB = R2R4;
BUFB = Y(RAR);
```

If desired, buffer input may be shifted right one bit as it crosses the channel (see field 17). The subcommand

LATCH;

shifts all future buffer memory input right one bit until the subcommand

UNLATCH;

is executed.

In ANIMIL programs, the 4 MSBs of the BARA and BARB registers are pointers to buffer memories and the 12 LSBs are pointers to addresses within the buffer memories. Buffers may be swapped (see field 17) by interchanging the high-order 4 bits of BARA and BARB with the subcommand

SWAP;

To prevent alteration of these buffer memory pointers (see field 25) in a microinstruction wherein BARA or BARB are destinations, the subcommand

HOLD;

is used.

The subcommand

INTERRUPT;

which instructs the IOCU to send an interrupt to the MCU, signals the end of a SPAU program (see field 27).

20. ABSTRACT

The SPE is a collection of several interconnected functional components. Two of these components, the Microprogrammed Control Unit (MCU) and the Signal Processing Arithmetic Unit (SPAU), are microprogrammable. This report describes the microprogramming support software developed at NRL to facilitate the development and debugging of microprograms for the Model XB-1 MCU and SPAU.

Microprogramming languages, AMIL for the MCU and ANIMIL for the SPAU, have been defined to facilitate microprogram creation. Translators have been developed for each language that convert the source languages to microcode for the MCU and SPAU. Microcode thus produced can be loaded into the control store of the associated machine.

In addition, simulators for the MCU and SPAU have been implemented to aid microprogram debugging and hardware testing. These simulators accept microcode produced by the translators and provide the user with convenient interactive control of the simulated machine. Both the MCU and SPAU simulators use the same user command language to control simulator execution and the display and modification of the simulated processor's registers.

Two ancillary programs have been written to aid the user in writing microprograms. A pre-processor provides translate-time macro facilities for both AMIL and ANIMIL, and a linkage editor allows programs to be written in modules and combined after translation.

20. ABSTRACT

The SPE is a collection of several interconnected functional components. Two of these components, the Microprogrammed Control Unit (MCU) and the Signal Processing Arithmetic Unit (SPAU), are microprogrammable. This report describes the microprogramming support software developed at NRL to facilitate the development and debugging of microprograms for the Model XB-1 MCU and SPAU.

Microprogramming languages, AMIL for the MCU and ANIMIL for the SPAU, have been defined to facilitate microprogram creation. Translators have been developed for each language that convert the source languages to microcode for the MCU and SPAU. Microcode thus produced can be loaded into the control store of the associated machine.

In addition, simulators for the MCU and SPAU have been implemented to aid microprogram debugging and hardware testing. These simulators accept microcode produced by the translators and provide the user with convenient interactive control of the simulated machine. Both the MCU and SPAU simulators use the same user command language to control simulator execution and the display and modification of the simulated processor's registers.

Two ancillary programs have been written to aid the user in writing microprograms. A preprocessor provides translate-time macro facilities for both AMIL and ANIMIL, and a linkage editor allows programs to be written in modules and combined after translation.

20. ABSTRACT

The SPE is a collection of several interconnected functional components. Two of these components, the Microprogrammed Control Unit (MCU) and the Signal Processing Arithmetic Unit (SPAU), are microprogrammable. This report describes the microprogramming support software developed at NRL to facilitate the development and debugging of microprograms for the Model XB-1 MCU and SPAU.

Microprogramming languages, AMIL for the MCU and ANIMIL for the SPAU, have been defined to facilitate microprogram creation. Translators have been developed for each language that convert the source languages to microcode for the MCU and SPAU. Microcode thus produced can be loaded into the control store of the associated machine.

In addition, simulators for the MCU and SPAU have been implemented to aid microprogram debugging and hardware testing. These simulators accept microcode produced by the translators and provide the user with convenient interactive control of the simulated machine. Both the MCU and SPAU simulators use the same user command language to control simulator execution and the display and modification of the simulated processor's registers.

Two ancillary programs have been written to aid the user in writing microprograms. A preprocessor provides translate-time macro facilities for both AMIL and ANIMIL, and a linkage editor allows programs to be written in modules and combined after translation.

20. ABSTRACT

The SPE is a collection of several interconnected functional components. Two of these components, the Microprogrammed Control Unit (MCU) and the Signal Processing Arithmetic Unit (SPAU), are microprogrammable. This report describes the microprogramming support software developed at NRL to facilitate the development and debugging of microprograms for the Model XB-1 MCU and SPAU.

Microprogramming languages, AMIL for the MCU and ANIMIL for the SPAU, have been defined to facilitate microprogram creation. Translators have been developed for each language that convert the source languages to microcode for the MCU and SPAU. Microcode thus produced can be loaded into the control store of the associated machine.

In addition, simulators for the MCU and SPAU have been implemented to aid microprogram debugging and hardware testing. These simulators accept microcode produced by the translators and provide the user with convenient interactive control of the simulated machine. Both the MCU and SPAU simulators use the same user command language to control simulator execution and the display and modification of the simulated processor's registers.

Two ancillary programs have been written to aid the user in writing microprograms. A preprocessor provides translate-time macro facilities for both AMIL and ANIMIL, and a linkage editor allows programs to be written in modules and combined after translation.

20. ABSTRACT

The SPE is a collection of several interconnected functional components. Two of these components, the Microprogrammed Control Unit (MCU) and the Signal Processing Arithmetic Unit (SPAU), are microprogrammable. This report describes the microprogramming support software developed at NRL to facilitate the development and debugging of microprograms for the Model XB-1 MCU and SPAU.

Microprogramming languages, AMIL for the MCU and ANIMIL for the SPAU, have been defined to facilitate microprogram creation. Translators have been developed for each language that convert the source languages to microcode for the MCU and SPAU. Microcode thus produced can be loaded into the control store of the associated machine.

In addition, simulators for the MCU and SPAU have been implemented to aid microprogram debugging and hardware testing. These simulators accept microcode produced by the translators and provide the user with convenient interactive control of the simulated machine. Both the MCU and SPAU simulators use the same user command language to control simulator execution and the display and modification of the simulated processor's registers.

Two ancillary programs have been written to aid the user in writing microprograms. A preprocessor provides translate-time macro facilities for both AMIL and ANIMIL, and a linkage editor allows programs to be written in modules and combined after translation.

20. ABSTRACT

The SPE is a collection of several interconnected functional components. Two of these components, the Microprogrammed Control Unit (MCU) and the Signal Processing Arithmetic Unit (SPAU), are microprogrammable. This report describes the microprogramming support software developed at NRL to facilitate the development and debugging of microprograms for the Model XB-1 MCU and SPAU.

Microprogramming languages, AMIL for the MCU and ANIMIL for the SPAU, have been defined to facilitate microprogram creation. Translators have been developed for each language that convert the source languages to microcode for the MCU and SPAU. Microcode thus produced can be loaded into the control store of the associated machine.

In addition, simulators for the MCU and SPAU have been implemented to aid microprogram debugging and hardware testing. These simulators accept microcode produced by the translators and provide the user with convenient interactive control of the simulated machine. Both the MCU and SPAU simulators use the same user command language to control simulator execution and the display and modification of the simulated processor's registers.

Two ancillary programs have been written to aid the user in writing microprograms. A pre-processor provides translate-time macro facilities for both AMIL and ANIMIL, and a linkage editor allows programs to be written in modules and combined after translation.

20. ABSTRACT

The SPE is a collection of several interconnected functional components. Two of these components, the Microprogrammed Control Unit (MCU) and the Signal Processing Arithmetic Unit (SPAU), are microprogrammable. This report describes the microprogramming support software developed at NRL to facilitate the development and debugging of microprograms for the Model XB-1 MCU and SPAU.

Microprogramming languages, AMIL for the MCU and ANIMIL for the SPAU, have been defined to facilitate microprogram creation. Translators have been developed for each language that convert the source languages to microcode for the MCU and SPAU. Microcode thus produced can be loaded into the control store of the associated machine.

In addition, simulators for the MCU and SPAU have been implemented to aid microprogram debugging and hardware testing. These simulators accept microcode produced by the translators and provide the user with convenient interactive control of the simulated machine. Both the MCU and SPAU simulators use the same user command language to control simulator execution and the display and modification of the simulated processor's registers.

Two ancillary programs have been written to aid the user in writing microprograms. A preprocessor provides translate-time macro facilities for both AMIL and ANIMIL, and a linkage editor allows programs to be written in modules and combined after translation.

20. ABSTRACT

The SPE is a collection of several interconnected functional components. Two of these components, the Microprogrammed Control Unit (MCU) and the Signal Processing Arithmetic Unit (SPAU), are microprogrammable. This report describes the microprogramming support software developed at NRL to facilitate the development and debugging of microprograms for the Model XB-1 MCU and SPAU.

Microprogramming languages, AMIL for the MCU and ANIMIL for the SPAU, have been defined to facilitate microprogram creation. Translators have been developed for each language that convert the source languages to microcode for the MCU and SPAU. Microcode thus produced can be loaded into the control store of the associated machine.

In addition, simulators for the MCU and SPAU have been implemented to aid microprogram debugging and hardware testing. These simulators accept microcode produced by the translators and provide the user with convenient interactive control of the simulated machine. Both the MCU and SPAU simulators use the same user command language to control simulator execution and the display and modification of the simulated processor's registers.

Two ancillary programs have been written to aid the user in writing microprograms. A preprocessor provides translate-time macro facilities for both AMIL and ANIMIL, and a linkage editor allows programs to be written in modules and combined after translation.

20. ABSTRACT

The SPE is a collection of several interconnected functional components. Two of these components, the Microprogrammed Control Unit (MCU) and the Signal Processing Arithmetic Unit (SPAU), are microprogrammable. This report describes the microprogramming support software developed at NRL to facilitate the development and debugging of microprograms for the Model XB-1 MCU and SPAU.

Microprogramming languages, AMIL for the MCU and ANIMIL for the SPAU, have been defined to facilitate microprogram creation. Translators have been developed for each language that convert the source languages to microcode for the MCU and SPAU. Microcode thus produced can be loaded into the control store of the associated machine.

In addition, simulators for the MCU and SPAU have been implemented to aid microprogram debugging and hardware testing. These simulators accept microcode produced by the translators and provide the user with convenient interactive control of the simulated machine. Both the MCU and SPAU simulators use the same user command language to control simulator execution and the display and modification of the simulated processor's registers.

Two ancillary programs have been written to aid the user in writing microprograms. A preprocessor provides translate-time macro facilities for both AMIL and ANIMIL, and a linkage editor allows programs to be written in modules and combined after translation.

```
## ERROR - INVALID SOURCE FOR X1, X2, Y1, OR Y2
## ERROR - INVALID X OR Y DESTINATION
## ERROR - INVALID SOURCE FOR X OR Y
## ERROR - INVALID SOURCE FOR LIT
## ERROR - INVALID COUNTER INPUT
## ERROR - COUNTER IS BOTH DECREMENTED AND USED AS A DESTINATION
## ERROR - INVALID BUFFER SOURCE
## ERROR - INVALID MULTIPLIER BIT SELECTION
## ERROR - INVALID HOLD OPERATION
## ERROR - INVALID INTERRUPT OPERATION
## ERROR - INVALID GO TO ADDRESS
## ERROR - DIFFERENT LABELS REFERENCED
## ERROR - INVALID LABEL
## ERROR - LABEL TABLE FULL
## ERROR - INVALID CONDITION
## ERROR - INVALID CLAUSE FOLLOWING CONDITION
## ERROR - INVALID CONTROL SEQUENCE SPECIFIED AFTER THEN
## ERROR - INVALID DECREMENT REGISTER SPECIFIED
## ERROR - INVALID ENT OR EXT INSTRUCTION
## ERROR - INVALID LIST OF LABELS IN ENT OR EXT INSTRUCTION
## ERROR - INVALID ORIG STATEMENT
## ERROR - TOO MANY ORIGIN STATEMENTS
## ERROR - INVALID SOURCE OR DESTINATION IN ASSIGNMENT STATEMENT
## ERROR - UNKNOWN INSTRUCTION TYPE
## ERROR - USE OF CONTROL FIELD - CONFLICTS WITH PREVIOUS USE

                           GROUP 2

## ERROR - INPUT STATEMENT LENGTH EXCEEDS 1400
           FIRST 1400 CHARACTERS ARE USED
           CHECK FOR UNMATCHED DOUBLE QUOTATION MARKS
## ERROR - LABEL WITH LENGTH 0 IS IGNORED
## ERROR - LABEL HAS MORE THAN 10 CHARACTERS
           FIRST 10 CHARACTERS ARE USED
```

Fig. 8—ANIMIL error messages (Continued)

20. ABSTRACT

The SPE is a collection of several interconnected functional components. Two of these components, the Microprogrammed Control Unit (MCU) and the Signal Processing Arithmetic Unit (SPAU), are microprogrammable. This report describes the microprogramming support software developed at NRL to facilitate the development and debugging of microprograms for the Model XB-1 MCU and SPAU.

Microprogramming languages, AMIL for the MCU and ANIMIL for the SPAU, have been defined to facilitate microprogram creation. Translators have been developed for each language that convert the source languages to microcode for the MCU and SPAU. Microcode thus produced can be loaded into the control store of the associated machine.

In addition, simulators for the MCU and SPAU have been implemented to aid microprogram debugging and hardware testing. These simulators accept microcode produced by the translators and provide the user with convenient interactive control of the simulated machine. Both the MCU and SPAU simulators use the same user command language to control simulator execution and the display and modification of the simulated processor's registers.

Two ancillary programs have been written to aid the user in writing microprograms. A preprocessor provides translate-time macro facilities for both AMIL and ANIMIL, and a linkage editor allows programs to be written in modules and combined after translation.

20. ABSTRACT

The SPE is a collection of several interconnected functional components. Two of these components, the Microprogrammed Control Unit (MCU) and the Signal Processing Arithmetic Unit (SPAU), are microprogrammable. This report describes the microprogramming support software developed at NRL to facilitate the development and debugging of microprograms for the Model XB-1 MCU and SPAU.

Microprogramming languages, AMIL for the MCU and ANIMIL for the SPAU, have been defined to facilitate microprogram creation. Translators have been developed for each language that convert the source languages to microcode for the MCU and SPAU. Microcode thus produced can be loaded into the control store of the associated machine.

In addition, simulators for the MCU and SPAU have been implemented to aid microprogram debugging and hardware testing. These simulators accept microcode produced by the translators and provide the user with convenient interactive control of the simulated machine. Both the MCU and SPAU simulators use the same user command language to control simulator execution and the display and modification of the simulated processor's registers.

Two ancillary programs have been written to aid the user in writing microprograms. A preprocessor provides translate-time macro facilities for both AMIL and ANIMIL, and a linkage editor allows programs to be written in modules and combined after translation.

7. A superscript + sign means that the item may be repeated. Repetition is zero or more times for elements and bracketed items and one or more times for items within braces.

Set and Display Commands

Set Command—The format of the set command is

SET facilityname = data [,facilityname = data]$^+$;

The names of the facilities comprise several groups: single registers, array registers, memory registers, and control store.

The MCU single registers are

ACSAR	
ADROV	
BARA	
BARB	
BUSA	
BUSB	
CARRY	
CSAR	
CTR	
CTROV	
DLSA	DLSA(DLSB) is the LSA(LSB) element addressed
DLSB	by the control store field DAAL(DBAL).
FSCR	
FSDR	
FSU	
LEAST	
LIT	
MOST	
SAR	
SCSAR	
SLSA	SLSA(SLSB) is the LSA(LSB) element addressed
SLSB	by the control store field SAAL(SBAL).
Z	
ZERO	

The MCU array registers are

LSA (subscript [-subscript])
LSB (subscript [-subscript])

Subscripts for array registers may range from 0 to 15.

The MCU memory registers are

$$\text{BUF}\begin{Bmatrix}0\\1\\2\\3\end{Bmatrix}(\underline{\text{halfwordnumber}}[-\underline{\text{halfwordnumber}}])$$

Buffer references are to half-words (16 bits), not to full words (32 bits). Even-numbered half-words, starting with zero, refer to the low-order 16 bits in consecutive words. The half-words in each buffer are referenced 0 to 8191.

The MCU control storage register is

$$\text{CS }(\underline{\text{address}}, \underline{\text{fieldnumber}} \ [-\underline{\text{fieldnumber}}])$$

The SPAU single registers are

ACSAR
BARA
BARB
CLOCK
CSAR
CTRI
CTRJ
CTRK
INCA
INCB
P1
P2
P3
P4
RAR
R1
R2
R3
R4
R7
Z1
Z2
Z3
Z4

The SPAU array registers are

COND (<u>subscript</u>)
W (<u>subscript</u>)
X1 (<u>subscript</u>)
X2 (<u>subscript</u>)
Y1 (<u>subscript</u>)
Y2 (<u>subscript</u>)

The subscript for array registers other than the W array may range from 0 to 15. The W array has a range of 0 to 31. Ranges of subscripts for array registers may be written

$$\underline{\text{subscript}} - \underline{\text{subscript}}$$

For example, X1(3-5). The elements of the COND array correspond to the 16 SPAU conditions (see field 28 in Appendix B).

The SPAU memory registers are

$$\text{BUF}\left\{\begin{matrix}0\\1\\2\\3\end{matrix}\right\}(\underline{\text{halfwordnumber}}[-\underline{\text{halfwordnumber}}])$$

Buffer references are to half-words (16 bits), not to full words (32 bits). Even-numbered half-words starting with zero refer to the high-order 16 bits in consecutive words. The half-words in each buffer are referenced in 0 through 8191.

$$\text{ROM }(\underline{\text{wordnumber}}[-\underline{\text{wordnumber}}])$$

ROM word numbers range from 0 to 2047. The first 1025 ROM words contain two 16-bit coefficients that are the sine and cosine of angles between 0 and $\pi/2$. These sine and cosine coefficients may not be altered. When read into the Z registers, the first (sine) coefficient is put into Z1 and the second (cosine) coefficient is put into Z2. The remaining 1023 coefficient store words contain four 16-bit coefficients that may be changed under microprogram control. When read into Z registers, the left-most (first) coefficient is put in Z1, the second coefficient is put in Z2, and so forth. Unlike other facilities, each reference to ROM indicates either two values (if the address is less than or equal to 1024) or four values (if the address is greater than 1024).

The SPAU control storage register is

$$\text{CS }(\underline{\text{address}}, \underline{\text{fieldnumber}}\ [-\underline{\text{fieldnumber}}])$$

For each item in the facility name of the SET command, there must be a corresponding item in the data list. Items in the data list are constants that may be specified directly in the input line or indirectly from a file. The form for constants is

$$[\pm]\ [\text{H}]\ [\,.\,]\ \underline{\text{integer}}$$

where H specifies a hexadecimal constant. The value of integer is limited by the precision of the facility. If the value specified is too large for the specified facility, an error message is printed and the user is asked to input another value. All numbers will be converted to 16-bit, two's complement binary representation. Decimal fractions will be multiplied by 32768 to yield 16-bit integers; hexadecimal fractions will be multiplied by 8000_{16}. Some examples follow:

Integer Representation	Stored Value
100	0064_{16}
–100	$FF9C_{16}$
+ .5	4000_{16}
- .5	$C000_{16}$
+H100	0100_{16}
-H100	$FF00_{16}$
+H.8	4000_{16}
–H.8	$C000_{16}$

In addition, the repetition of a single number can be specified by

# repetitionnumber (number)

For example,

Representation	Meaning
#3(0)	0,0,0
#5(H.1)	2048,2048,2048,2048,2048
#2(–.5)	49152,49152

For frequently referenced data, an external file can be used as the source of data items:

&filename

Data items will be read from the file with blanks or commas serving as delimiters. Extra data items in a file are ignored. Providing insufficient items is an error. Some examples of the SET command for the MCU are the following:

```
SET BARA = 100
S LSB(1) = 0
S LSA(0-13) = 0,1,2,#11(H.5);
```

Some examples for the SPAU include the following:

```
SET BARA = 0;
SET CTRI = 0, CTRJ = 1;
S R1 = .4, R2 = H.25, R3 = -H.25, R4 = HFFF;
S W (0-31) = 1,2, # 30(0);
  "SETS W(0)=1, W(1)=2, W(2)=0, W(3)=0, ... W(31)=0"
S BUF0 (0-8091) = #4095(H.C), 0,&BUFIN;
  "SETS FIRST 4095 HALFWORDS IN"
  "BUFFER 0 TO 24576, THE NEXT HALFWORD"
  "TO ZERO, AND THE REMAINING HALFWORDS"
  "ARE ASSIGNED FROM CONSECUTIVE ENTRIES OF FILE BUFIN"
```

20. ABSTRACT

The SPE is a collection of several interconnected functional components. Two of these components, the Microprogrammed Control Unit (MCU) and the Signal Processing Arithmetic Unit (SPAU), are microprogrammable. This report describes the microprogramming support software developed at NRL to facilitate the development and debugging of microprograms for the Model XB-1 MCU and SPAU.

Microprogramming languages, AMIL for the MCU and ANIMIL for the SPAU, have been defined to facilitate microprogram creation. Translators have been developed for each language that convert the source languages to microcode for the MCU and SPAU. Microcode thus produced can be loaded into the control store of the associated machine.

In addition, simulators for the MCU and SPAU have been implemented to aid microprogram debugging and hardware testing. These simulators accept microcode produced by the translators and provide the user with convenient interactive control of the simulated machine. Both the MCU and SPAU simulators use the same user command language to control simulator execution and the display and modification of the simulated processor's registers.

Two ancillary programs have been written to aid the user in writing microprograms. A preprocessor provides translate-time macro facilities for both AMIL and ANIMIL, and a linkage editor allows programs to be written in modules and combined after translation.

20. ABSTRACT

The SPE is a collection of several interconnected functional components. Two of these components, the Microprogrammed Control Unit (MCU) and the Signal Processing Arithmetic Unit (SPAU), are microprogrammable. This report describes the microprogramming support software developed at NRL to facilitate the development and debugging of microprograms for the Model XB-1 MCU and SPAU.

Microprogramming languages, AMIL for the MCU and ANIMIL for the SPAU, have been defined to facilitate microprogram creation. Translators have been developed for each language that convert the source languages to microcode for the MCU and SPAU. Microcode thus produced can be loaded into the control store of the associated machine.

In addition, simulators for the MCU and SPAU have been implemented to aid microprogram debugging and hardware testing. These simulators accept microcode produced by the translators and provide the user with convenient interactive control of the simulated machine. Both the MCU and SPAU simulators use the same user command language to control simulator execution and the display and modification of the simulated processor's registers.

Two ancillary programs have been written to aid the user in writing microprograms. A preprocessor provides translate-time macro facilities for both AMIL and ANIMIL, and a linkage editor allows programs to be written in modules and combined after translation.

20. ABSTRACT

The SPE is a collection of several interconnected functional components. Two of these components, the Microprogrammed Control Unit (MCU) and the Signal Processing Arithmetic Unit (SPAU), are microprogrammable. This report describes the microprogramming support software developed at NRL to facilitate the development and debugging of microprograms for the Model XB-1 MCU and SPAU.

Microprogramming languages, AMIL for the MCU and ANIMIL for the SPAU, have been defined to facilitate microprogram creation. Translators have been developed for each language that convert the source languages to microcode for the MCU and SPAU. Microcode thus produced can be loaded into the control store of the associated machine.

In addition, simulators for the MCU and SPAU have been implemented to aid microprogram debugging and hardware testing. These simulators accept microcode produced by the translators and provide the user with convenient interactive control of the simulated machine. Both the MCU and SPAU simulators use the same user command language to control simulator execution and the display and modification of the simulated processor's registers.

Two ancillary programs have been written to aid the user in writing microprograms. A preprocessor provides translate-time macro facilities for both AMIL and ANIMIL, and a linkage editor allows programs to be written in modules and combined after translation.

Remove Trace Range and Remove Break Range Commands—The commands

REMOVE TRACE RANGE csaddr$_1$-csaddr$_2$ [,csaddr$_1$-csaddr$_2$]$^+$;
REMOVE BREAK RANGE csaddr$_1$-csaddr$_2$ [,csaddr$_1$-csaddr$_2$]$^+$;

mean that control store addresses between each pair of csaddr$_1$ and csaddr$_2$ inclusive are no longer trace points or breakpoints.

Clear Traces and Clear Breaks Commands—These commands,

CLEAR TRACES;
CLEAR BREAKS;

mean there are no longer any trace points or breakpoints, whichever is specified. The command

CLEAR;

means there are no longer any trace points or breakpoints. Because trace point and breakpoint information is stored with the control store, an object program must be loaded before trace and break commands may be executed. Additional commands facilitate specification of trace points and breakpoints, depending on the number of clock cycles in the simulation.

Interval Command—The command

INTERVAL integer;

means that trace information should be printed every integer cycles where integer is an unsigned decimal number. This command does not affect previously defined trace points or breakpoints for which information will be printed as usual.

Single Step Mode Command—This command,

SINGLE STEP MODE;

effectively establishes a breakpoint at every step (i.e., every clock cycle) in the simulation. Although this command has the effect of making every control store address a breakpoint, it does not change previously specified breakpoints.

No Step Command—The command

NO STEP;

exists single step mode and reestablishes previous breakpoint definitions.

Some examples of trace point and breakpoint commands for the MCU are as follows:

20. ABSTRACT

The SPE is a collection of several interconnected functional components. Two of these components, the Microprogrammed Control Unit (MCU) and the Signal Processing Arithmetic Unit (SPAU), are microprogrammable. This report describes the microprogramming support software developed at NRL to facilitate the development and debugging of microprograms for the Model XB-1 MCU and SPAU.

Microprogramming languages, AMIL for the MCU and ANIMIL for the SPAU, have been defined to facilitate microprogram creation. Translators have been developed for each language that convert the source languages to microcode for the MCU and SPAU. Microcode thus produced can be loaded into the control store of the associated machine.

In addition, simulators for the MCU and SPAU have been implemented to aid microprogram debugging and hardware testing. These simulators accept microcode produced by the translators and provide the user with convenient interactive control of the simulated machine. Both the MCU and SPAU simulators use the same user command language to control simulator execution and the display and modification of the simulated processor's registers.

Two ancillary programs have been written to aid the user in writing microprograms. A pre-processor provides translate-time macro facilities for both AMIL and ANIMIL, and a linkage editor allows programs to be written in modules and combined after translation.

Using the Simulators and Command Language Interpreter

The first versions of the MCU and SPAU simulators with the command language interpreter have been developed on the KRONOS time-sharing system. To use the simulators, sign on the system, create an object program by using the AMIL or ANIMIL translator or the linkage editor and enter the command

```
EXECUTE
```

to which the system will respond

```
READY.
```

Enter the command

```
ATTACH,DMCU/UN=K5037WU
```

to indicate MCU simulation or the command

```
ATTACH,DSPAU/UN=K5037WU
```

to indicate SPAU simulation. Enter the command

```
RUN,I=DMCU
```

to initiate MCU simulation or

```
RUN,I=DSPAU
```

to initiate SPAU simulation. The command language interpreter will print

```
 MCU COMMAND PROCESSOR
?
```

or

```
 SPAU COMMAND PROCESSOR
?
```

depending on the machine being simulated. At this point the user may enter commands to load the program to be simulated, initialize registers, and indicate trace points and breakpoints. A GO command initiates simulation, which continues until a breakpoint or the end of the microprogram is reached. At these points, the user may again enter commands. The END command terminates the run.

Error Messages

In interpreting the input commands, the CLI may discover errors. Upon encountering an error, the CLI prints one or more messages that indicate the error, prints the command that it ignores because of the error, and continues processing with the next command. If a line with a GO command contains errors, the GO command is not executed so that corrections can be made before initiating simulation. In addition, the simulators

20. ABSTRACT

The SPE is a collection of several interconnected functional components. Two of these components, the Microprogrammed Control Unit (MCU) and the Signal Processing Arithmetic Unit (SPAU), are microprogrammable. This report describes the microprogramming support software developed at NRL to facilitate the development and debugging of microprograms for the Model XB-1 MCU and SPAU.

Microprogramming languages, AMIL for the MCU and ANIMIL for the SPAU, have been defined to facilitate microprogram creation. Translators have been developed for each language that convert the source languages to microcode for the MCU and SPAU. Microcode thus produced can be loaded into the control store of the associated machine.

In addition, simulators for the MCU and SPAU have been implemented to aid microprogram debugging and hardware testing. These simulators accept microcode produced by the translators and provide the user with convenient interactive control of the simulated machine. Both the MCU and SPAU simulators use the same user command language to control simulator execution and the display and modification of the simulated processor's registers.

Two ancillary programs have been written to aid the user in writing microprograms. A preprocessor provides translate-time macro facilities for both AMIL and ANIMIL, and a linkage editor allows programs to be written in modules and combined after translation.

File Commands

5. LOAD $\underline{\text{filename}}$;

$$\text{6. REPLACE } \underline{\text{filename}} = \left\{ \begin{matrix} \text{BUF0} \\ \text{BUF1} \\ \text{BUF2} \\ \text{BUF3} \\ \text{CS} \\ \text{ROM} \end{matrix} \right\}$$

Output Control Commands

7. $\underline{\text{A}}\text{DD } \underline{\text{facilityname}}\ [=\underline{\text{format}}]\ [,\underline{\text{facilityname}}[=\underline{\text{format}}]]^{+}$;

8. $\underline{\text{D}}\text{ELETE } \underline{\text{entrynumber}}\ [,\underline{\text{entrynumber}}]^{+}$;

Trace and Break Commands

$$\begin{matrix} \text{9.} \\ \\ \text{10.} \end{matrix} \left\{ \begin{matrix} \underline{\text{T}}\text{RACE} \\ \\ \underline{\text{B}}\text{REAK} \end{matrix} \right\} \underline{\text{csaddr}}\ [,\underline{\text{csaddr}}]^{+};$$

$$\begin{matrix} \text{11.} \\ \\ \text{12.} \end{matrix} \left\{ \begin{matrix} \underline{\text{T}}\text{RACE} \\ \\ \underline{\text{B}}\text{REAK} \end{matrix} \right\} \underline{\text{R}}\text{ANGE } \underline{\text{csaddr}}_1 - \underline{\text{csaddr}}_2\ [,\underline{\text{csaddr}}_1 - \underline{\text{csaddr}}_2]^{+};$$

$$\begin{matrix} \text{13.} \\ \\ \text{14.} \end{matrix} \ \underline{\text{R}}\text{EMOVE} \left\{ \begin{matrix} \underline{\text{T}}\text{RACE} \\ \\ \underline{\text{B}}\text{REAK} \end{matrix} \right\} \underline{\text{csaddr}}\ [,\underline{\text{csaddr}}]^{+};$$

$$\begin{matrix} \text{15.} \\ \\ \text{16.} \end{matrix} \ \underline{\text{R}}\text{EMOVE} \left\{ \begin{matrix} \underline{\text{T}}\text{RACE} \\ \\ \underline{\text{B}}\text{REAK} \end{matrix} \right\} \underline{\text{R}}\text{ANGE } \underline{\text{csaddr}}_1 - \underline{\text{csaddr}}_2\ [,\underline{\text{csaddr}}_1 - \underline{\text{csaddr}}_2]^{+};$$

$$\begin{matrix} \text{17.} \\ \\ \text{18.} \end{matrix} \ \underline{\text{C}}\text{LEAR} \left\{ \begin{matrix} \underline{\text{T}}\text{RACES} \\ \\ \underline{\text{B}}\text{REAKS} \end{matrix} \right\};$$

19. $\underline{\text{C}}$LEAR;

20. $\underline{\text{I}}$NTERVAL $\underline{\text{integer}}$;

21. $\underline{\text{S}}$INGLE $\underline{\text{S}}$TEP $\underline{\text{M}}$ODE;

22. $\underline{\text{N}}$O $\underline{\text{S}}$TEP;

Input Control Command

23. $\underline{\text{F}}$ILE $\underline{\text{I}}$NPUT filename;

20. ABSTRACT

The SPE is a collection of several interconnected functional components. Two of these components, the Microprogrammed Control Unit (MCU) and the Signal Processing Arithmetic Unit (SPAU), are microprogrammable. This report describes the microprogramming support software developed at NRL to facilitate the development and debugging of microprograms for the Model XB-1 MCU and SPAU.

Microprogramming languages, AMIL for the MCU and ANIMIL for the SPAU, have been defined to facilitate microprogram creation. Translators have been developed for each language that convert the source languages to microcode for the MCU and SPAU. Microcode thus produced can be loaded into the control store of the associated machine.

In addition, simulators for the MCU and SPAU have been implemented to aid microprogram debugging and hardware testing. These simulators accept microcode produced by the translators and provide the user with convenient interactive control of the simulated machine. Both the MCU and SPAU simulators use the same user command language to control simulator execution and the display and modification of the simulated processor's registers.

Two ancillary programs have been written to aid the user in writing microprograms. A preprocessor provides translate-time macro facilities for both AMIL and ANIMIL, and a linkage editor allows programs to be written in modules and combined after translation.

20. ABSTRACT

The SPE is a collection of several interconnected functional components. Two of these components, the Microprogrammed Control Unit (MCU) and the Signal Processing Arithmetic Unit (SPAU), are microprogrammable. This report describes the microprogramming support software developed at NRL to facilitate the development and debugging of microprograms for the Model XB-1 MCU and SPAU.

Microprogramming languages, AMIL for the MCU and ANIMIL for the SPAU, have been defined to facilitate microprogram creation. Translators have been developed for each language that convert the source languages to microcode for the MCU and SPAU. Microcode thus produced can be loaded into the control store of the associated machine.

In addition, simulators for the MCU and SPAU have been implemented to aid microprogram debugging and hardware testing. These simulators accept microcode produced by the translators and provide the user with convenient interactive control of the simulated machine. Both the MCU and SPAU simulators use the same user command language to control simulator execution and the display and modification of the simulated processor's registers.

Two ancillary programs have been written to aid the user in writing microprograms. A preprocessor provides translate-time macro facilities for both AMIL and ANIMIL, and a linkage editor allows programs to be written in modules and combined after translation.

20. ABSTRACT

The SPE is a collection of several interconnected functional components. Two of these components, the Microprogrammed Control Unit (MCU) and the Signal Processing Arithmetic Unit (SPAU), are microprogrammable. This report describes the microprogramming support software developed at NRL to facilitate the development and debugging of microprograms for the Model XB-1 MCU and SPAU.

Microprogramming languages, AMIL for the MCU and ANIMIL for the SPAU, have been defined to facilitate microprogram creation. Translators have been developed for each language that convert the source languages to microcode for the MCU and SPAU. Microcode thus produced can be loaded into the control store of the associated machine.

In addition, simulators for the MCU and SPAU have been implemented to aid microprogram debugging and hardware testing. These simulators accept microcode produced by the translators and provide the user with convenient interactive control of the simulated machine. Both the MCU and SPAU simulators use the same user command language to control simulator execution and the display and modification of the simulated processor's registers.

Two ancillary programs have been written to aid the user in writing microprograms. A preprocessor provides translate-time macro facilities for both AMIL and ANIMIL, and a linkage editor allows programs to be written in modules and combined after translation.

20. ABSTRACT

The SPE is a collection of several interconnected functional components. Two of these components, the Microprogrammed Control Unit (MCU) and the Signal Processing Arithmetic Unit (SPAU), are microprogrammable. This report describes the microprogramming support software developed at NRL to facilitate the development and debugging of microprograms for the Model XB-1 MCU and SPAU.

Microprogramming languages, AMIL for the MCU and ANIMIL for the SPAU, have been defined to facilitate microprogram creation. Translators have been developed for each language that convert the source languages to microcode for the MCU and SPAU. Microcode thus produced can be loaded into the control store of the associated machine.

In addition, simulators for the MCU and SPAU have been implemented to aid microprogram debugging and hardware testing. These simulators accept microcode produced by the translators and provide the user with convenient interactive control of the simulated machine. Both the MCU and SPAU simulators use the same user command language to control simulator execution and the display and modification of the simulated processor's registers.

Two ancillary programs have been written to aid the user in writing microprograms. A preprocessor provides translate-time macro facilities for both AMIL and ANIMIL, and a linkage editor allows programs to be written in modules and combined after translation.

20. ABSTRACT

The SPE is a collection of several interconnected functional components. Two of these components, the Microprogrammed Control Unit (MCU) and the Signal Processing Arithmetic Unit (SPAU), are microprogrammable. This report describes the microprogramming support software developed at NRL to facilitate the development and debugging of microprograms for the Model XB-1 MCU and SPAU.

Microprogramming languages, AMIL for the MCU and ANIMIL for the SPAU, have been defined to facilitate microprogram creation. Translators have been developed for each language that convert the source languages to microcode for the MCU and SPAU. Microcode thus produced can be loaded into the control store of the associated machine.

In addition, simulators for the MCU and SPAU have been implemented to aid microprogram debugging and hardware testing. These simulators accept microcode produced by the translators and provide the user with convenient interactive control of the simulated machine. Both the MCU and SPAU simulators use the same user command language to control simulator execution and the display and modification of the simulated processor's registers.

Two ancillary programs have been written to aid the user in writing microprograms. A preprocessor provides translate-time macro facilities for both AMIL and ANIMIL, and a linkage editor allows programs to be written in modules and combined after translation.

20. ABSTRACT

The SPE is a collection of several interconnected functional components. Two of these components, the Microprogrammed Control Unit (MCU) and the Signal Processing Arithmetic Unit (SPAU), are microprogrammable. This report describes the microprogramming support software developed at NRL to facilitate the development and debugging of microprograms for the Model XB-1 MCU and SPAU.

Microprogramming languages, AMIL for the MCU and ANIMIL for the SPAU, have been defined to facilitate microprogram creation. Translators have been developed for each language that convert the source languages to microcode for the MCU and SPAU. Microcode thus produced can be loaded into the control store of the associated machine.

In addition, simulators for the MCU and SPAU have been implemented to aid microprogram debugging and hardware testing. These simulators accept microcode produced by the translators and provide the user with convenient interactive control of the simulated machine. Both the MCU and SPAU simulators use the same user command language to control simulator execution and the display and modification of the simulated processor's registers.

Two ancillary programs have been written to aid the user in writing microprograms. A preprocessor provides translate-time macro facilities for both AMIL and ANIMIL, and a linkage editor allows programs to be written in modules and combined after translation.

20. ABSTRACT

The SPE is a collection of several interconnected functional components. Two of these components, the Microprogrammed Control Unit (MCU) and the Signal Processing Arithmetic Unit (SPAU), are microprogrammable. This report describes the microprogramming support software developed at NRL to facilitate the development and debugging of microprograms for the Model XB-1 MCU and SPAU.

Microprogramming languages, AMIL for the MCU and ANIMIL for the SPAU, have been defined to facilitate microprogram creation. Translators have been developed for each language that convert the source languages to microcode for the MCU and SPAU. Microcode thus produced can be loaded into the control store of the associated machine.

In addition, simulators for the MCU and SPAU have been implemented to aid microprogram debugging and hardware testing. These simulators accept microcode produced by the translators and provide the user with convenient interactive control of the simulated machine. Both the MCU and SPAU simulators use the same user command language to control simulator execution and the display and modification of the simulated processor's registers.

Two ancillary programs have been written to aid the user in writing microprograms. A preprocessor provides translate-time macro facilities for both AMIL and ANIMIL, and a linkage editor allows programs to be written in modules and combined after translation.

20. ABSTRACT

The SPE is a collection of several interconnected functional components. Two of these components, the Microprogrammed Control Unit (MCU) and the Signal Processing Arithmetic Unit (SPAU), are microprogrammable. This report describes the microprogramming support software developed at NRL to facilitate the development and debugging of microprograms for the Model XB-1 MCU and SPAU.

Microprogramming languages, AMIL for the MCU and ANIMIL for the SPAU, have been defined to facilitate microprogram creation. Translators have been developed for each language that convert the source languages to microcode for the MCU and SPAU. Microcode thus produced can be loaded into the control store of the associated machine.

In addition, simulators for the MCU and SPAU have been implemented to aid microprogram debugging and hardware testing. These simulators accept microcode produced by the translators and provide the user with convenient interactive control of the simulated machine. Both the MCU and SPAU simulators use the same user command language to control simulator execution and the display and modification of the simulated processor's registers.

Two ancillary programs have been written to aid the user in writing microprograms. A preprocessor provides translate-time macro facilities for both AMIL and ANIMIL, and a linkage editor allows programs to be written in modules and combined after translation.

20. ABSTRACT

The SPE is a collection of several interconnected functional components. Two of these components, the Microprogrammed Control Unit (MCU) and the Signal Processing Arithmetic Unit (SPAU), are microprogrammable. This report describes the microprogramming support software developed at NRL to facilitate the development and debugging of microprograms for the Model XB-1 MCU and SPAU.

Microprogramming languages, AMIL for the MCU and ANIMIL for the SPAU, have been defined to facilitate microprogram creation. Translators have been developed for each language that convert the source languages to microcode for the MCU and SPAU. Microcode thus produced can be loaded into the control store of the associated machine.

In addition, simulators for the MCU and SPAU have been implemented to aid microprogram debugging and hardware testing. These simulators accept microcode produced by the translators and provide the user with convenient interactive control of the simulated machine. Both the MCU and SPAU simulators use the same user command language to control simulator execution and the display and modification of the simulated processor's registers.

Two ancillary programs have been written to aid the user in writing microprograms. A preprocessor provides translate-time macro facilities for both AMIL and ANIMIL, and a linkage editor allows programs to be written in modules and combined after translation.

20. ABSTRACT

The SPE is a collection of several interconnected functional components. Two of these components, the Microprogrammed Control Unit (MCU) and the Signal Processing Arithmetic Unit (SPAU), are microprogrammable. This report describes the microprogramming support software developed at NRL to facilitate the development and debugging of microprograms for the Model XB-1 MCU and SPAU.

Microprogramming languages, AMIL for the MCU and ANIMIL for the SPAU, have been defined to facilitate microprogram creation. Translators have been developed for each language that convert the source languages to microcode for the MCU and SPAU. Microcode thus produced can be loaded into the control store of the associated machine.

In addition, simulators for the MCU and SPAU have been implemented to aid microprogram debugging and hardware testing. These simulators accept microcode produced by the translators and provide the user with convenient interactive control of the simulated machine. Both the MCU and SPAU simulators use the same user command language to control simulator execution and the display and modification of the simulated processor's registers.

Two ancillary programs have been written to aid the user in writing microprograms. A preprocessor provides translate-time macro facilities for both AMIL and ANIMIL, and a linkage editor allows programs to be written in modules and combined after translation.

20. ABSTRACT

The SPE is a collection of several interconnected functional components. Two of these components, the Microprogrammed Control Unit (MCU) and the Signal Processing Arithmetic Unit (SPAU), are microprogrammable. This report describes the microprogramming support software developed at NRL to facilitate the development and debugging of microprograms for the Model XB-1 MCU and SPAU.

Microprogramming languages, AMIL for the MCU and ANIMIL for the SPAU, have been defined to facilitate microprogram creation. Translators have been developed for each language that convert the source languages to microcode for the MCU and SPAU. Microcode thus produced can be loaded into the control store of the associated machine.

In addition, simulators for the MCU and SPAU have been implemented to aid microprogram debugging and hardware testing. These simulators accept microcode produced by the translators and provide the user with convenient interactive control of the simulated machine. Both the MCU and SPAU simulators use the same user command language to control simulator execution and the display and modification of the simulated processor's registers.

Two ancillary programs have been written to aid the user in writing microprograms. A preprocessor provides translate-time macro facilities for both AMIL and ANIMIL, and a linkage editor allows programs to be written in modules and combined after translation.

Examples

Figure 11 illustrates use of the preprocessor with an AMIL program.

Figure 12 shows a sample run of the ANIMIL translator with the preprocessor option; note the macros for subcommands, named MULT2, INC-BARA, MULT1, and ADD1-ADD3; the assignment of the parameter BUFARA to a long string; and the included text.

Errors

Errors may be detected by the preprocessor when processing specified file names and optional parameters or when preprocessing input text.

The error messages that may be printed during the processing of file names and optional parameters are as follows:

```
## ERROR - INVALID SOURCE FILE NAME
## ERROR - INVALID OBJECT FILE NAME
## ERROR - INVALID OPTIONAL PARAMETER
## ERROR - INVALID SOURCE MARGIN PARAMETER
## ERROR - INVALID LINELENGTH OPTION
```

After the processor prints an error message, it repeats its request for information

Figure 13 shows the error messages that may be printed by the preprocessor. Messages are printed after the line on which the error occurred. Following each message in the first group, the preprocessor prints

```
## ERROR - INVALID PREPROCESSOR STATEMENT
           SCAN RESUMED AFTER $ OR AT BEGINNING OF NEXT LINE
```

The preprocessor then ignores the statement in error up to the dollar sign or the end of the line, whichever comes first. The messages in the second group are self-explanatory.

Proposed Preprocessor Extensions

Although the preprocessor statements discussed provide general equivalence and macro facilities, additional facilities would be useful and would permit more flexibility. Those ideas may be implemented in later versions. The ability to have integer preprocessor variables and a general conditional statement of the form

```
& IF condition
    & THEN & preprocessor-statement $
    & ELSE & preprocessor-statement $
```

20. ABSTRACT

The SPE is a collection of several interconnected functional components. Two of these components, the Microprogrammed Control Unit (MCU) and the Signal Processing Arithmetic Unit (SPAU), are microprogrammable. This report describes the microprogramming support software developed at NRL to facilitate the development and debugging of microprograms for the Model XB-1 MCU and SPAU.

Microprogramming languages, AMIL for the MCU and ANIMIL for the SPAU, have been defined to facilitate microprogram creation. Translators have been developed for each language that convert the source languages to microcode for the MCU and SPAU. Microcode thus produced can be loaded into the control store of the associated machine.

In addition, simulators for the MCU and SPAU have been implemented to aid microprogram debugging and hardware testing. These simulators accept microcode produced by the translators and provide the user with convenient interactive control of the simulated machine. Both the MCU and SPAU simulators use the same user command language to control simulator execution and the display and modification of the simulated processor's registers.

Two ancillary programs have been written to aid the user in writing microprograms. A preprocessor provides translate-time macro facilities for both AMIL and ANIMIL, and a linkage editor allows programs to be written in modules and combined after translation.

20. ABSTRACT

The SPE is a collection of several interconnected functional components. Two of these components, the Microprogrammed Control Unit (MCU) and the Signal Processing Arithmetic Unit (SPAU), are microprogrammable. This report describes the microprogramming support software developed at NRL to facilitate the development and debugging of microprograms for the Model XB-1 MCU and SPAU.

Microprogramming languages, AMIL for the MCU and ANIMIL for the SPAU, have been defined to facilitate microprogram creation. Translators have been developed for each language that convert the source languages to microcode for the MCU and SPAU. Microcode thus produced can be loaded into the control store of the associated machine.

In addition, simulators for the MCU and SPAU have been implemented to aid microprogram debugging and hardware testing. These simulators accept microcode produced by the translators and provide the user with convenient interactive control of the simulated machine. Both the MCU and SPAU simulators use the same user command language to control simulator execution and the display and modification of the simulated processor's registers.

Two ancillary programs have been written to aid the user in writing microprograms. A preprocessor provides translate-time macro facilities for both AMIL and ANIMIL, and a linkage editor allows programs to be written in modules and combined after translation.

20. ABSTRACT

The SPE is a collection of several interconnected functional components. Two of these components, the Microprogrammed Control Unit (MCU) and the Signal Processing Arithmetic Unit (SPAU), are microprogrammable. This report describes the microprogramming support software developed at NRL to facilitate the development and debugging of microprograms for the Model XB-1 MCU and SPAU.

Microprogramming languages, AMIL for the MCU and ANIMIL for the SPAU, have been defined to facilitate microprogram creation. Translators have been developed for each language that convert the source languages to microcode for the MCU and SPAU. Microcode thus produced can be loaded into the control store of the associated machine.

In addition, simulators for the MCU and SPAU have been implemented to aid microprogram debugging and hardware testing. These simulators accept microcode produced by the translators and provide the user with convenient interactive control of the simulated machine. Both the MCU and SPAU simulators use the same user command language to control simulator execution and the display and modification of the simulated processor's registers.

Two ancillary programs have been written to aid the user in writing microprograms. A preprocessor provides translate-time macro facilities for both AMIL and ANIMIL, and a linkage editor allows programs to be written in modules and combined after translation.

20. ABSTRACT

The SPE is a collection of several interconnected functional components. Two of these components, the Microprogrammed Control Unit (MCU) and the Signal Processing Arithmetic Unit (SPAU), are microprogrammable. This report describes the microprogramming support software developed at NRL to facilitate the development and debugging of microprograms for the Model XB-1 MCU and SPAU.

Microprogramming languages, AMIL for the MCU and ANIMIL for the SPAU, have been defined to facilitate microprogram creation. Translators have been developed for each language that convert the source languages to microcode for the MCU and SPAU. Microcode thus produced can be loaded into the control store of the associated machine.

In addition, simulators for the MCU and SPAU have been implemented to aid microprogram debugging and hardware testing. These simulators accept microcode produced by the translators and provide the user with convenient interactive control of the simulated machine. Both the MCU and SPAU simulators use the same user command language to control simulator execution and the display and modification of the simulated processor's registers.

Two ancillary programs have been written to aid the user in writing microprograms. A pre-processor provides translate-time macro facilities for both AMIL and ANIMIL, and a linkage editor allows programs to be written in modules and combined after translation.

20. ABSTRACT

The SPE is a collection of several interconnected functional components. Two of these components, the Microprogrammed Control Unit (MCU) and the Signal Processing Arithmetic Unit (SPAU), are microprogrammable. This report describes the microprogramming support software developed at NRL to facilitate the development and debugging of microprograms for the Model XB-1 MCU and SPAU.

Microprogramming languages, AMIL for the MCU and ANIMIL for the SPAU, have been defined to facilitate microprogram creation. Translators have been developed for each language that convert the source languages to microcode for the MCU and SPAU. Microcode thus produced can be loaded into the control store of the associated machine.

In addition, simulators for the MCU and SPAU have been implemented to aid microprogram debugging and hardware testing. These simulators accept microcode produced by the translators and provide the user with convenient interactive control of the simulated machine. Both the MCU and SPAU simulators use the same user command language to control simulator execution and the display and modification of the simulated processor's registers.

Two ancillary programs have been written to aid the user in writing microprograms. A preprocessor provides translate-time macro facilities for both AMIL and ANIMIL, and a linkage editor allows programs to be written in modules and combined after translation.

20. ABSTRACT

The SPE is a collection of several interconnected functional components. Two of these components, the Microprogrammed Control Unit (MCU) and the Signal Processing Arithmetic Unit (SPAU), are microprogrammable. This report describes the microprogramming support software developed at NRL to facilitate the development and debugging of microprograms for the Model XB-1 MCU and SPAU.

Microprogramming languages, AMIL for the MCU and ANIMIL for the SPAU, have been defined to facilitate microprogram creation. Translators have been developed for each language that convert the source languages to microcode for the MCU and SPAU. Microcode thus produced can be loaded into the control store of the associated machine.

In addition, simulators for the MCU and SPAU have been implemented to aid microprogram debugging and hardware testing. These simulators accept microcode produced by the translators and provide the user with convenient interactive control of the simulated machine. Both the MCU and SPAU simulators use the same user command language to control simulator execution and the display and modification of the simulated processor's registers.

Two ancillary programs have been written to aid the user in writing microprograms. A preprocessor provides translate-time macro facilities for both AMIL and ANIMIL, and a linkage editor allows programs to be written in modules and combined after translation.

20. ABSTRACT

The SPE is a collection of several interconnected functional components. Two of these components, the Microprogrammed Control Unit (MCU) and the Signal Processing Arithmetic Unit (SPAU), are microprogrammable. This report describes the microprogramming support software developed at NRL to facilitate the development and debugging of microprograms for the Model XB-1 MCU and SPAU.

Microprogramming languages, AMIL for the MCU and ANIMIL for the SPAU, have been defined to facilitate microprogram creation. Translators have been developed for each language that convert the source languages to microcode for the MCU and SPAU. Microcode thus produced can be loaded into the control store of the associated machine.

In addition, simulators for the MCU and SPAU have been implemented to aid microprogram debugging and hardware testing. These simulators accept microcode produced by the translators and provide the user with convenient interactive control of the simulated machine. Both the MCU and SPAU simulators use the same user command language to control simulator execution and the display and modification of the simulated processor's registers.

Two ancillary programs have been written to aid the user in writing microprograms. A preprocessor provides translate-time macro facilities for both AMIL and ANIMIL, and a linkage editor allows programs to be written in modules and combined after translation.

20. ABSTRACT

The SPE is a collection of several interconnected functional components. Two of these components, the Microprogrammed Control Unit (MCU) and the Signal Processing Arithmetic Unit (SPAU), are microprogrammable. This report describes the microprogramming support software developed at NRL to facilitate the development and debugging of microprograms for the Model XB-1 MCU and SPAU.

Microprogramming languages, AMIL for the MCU and ANIMIL for the SPAU, have been defined to facilitate microprogram creation. Translators have been developed for each language that convert the source languages to microcode for the MCU and SPAU. Microcode thus produced can be loaded into the control store of the associated machine.

In addition, simulators for the MCU and SPAU have been implemented to aid microprogram debugging and hardware testing. These simulators accept microcode produced by the translators and provide the user with convenient interactive control of the simulated machine. Both the MCU and SPAU simulators use the same user command language to control simulator execution and the display and modification of the simulated processor's registers.

Two ancillary programs have been written to aid the user in writing microprograms. A preprocessor provides translate-time macro facilities for both AMIL and ANIMIL, and a linkage editor allows programs to be written in modules and combined after translation.

20. ABSTRACT

The SPE is a collection of several interconnected functional components. Two of these components, the Microprogrammed Control Unit (MCU) and the Signal Processing Arithmetic Unit (SPAU), are microprogrammable. This report describes the microprogramming support software developed at NRL to facilitate the development and debugging of microprograms for the Model XB-1 MCU and SPAU.

Microprogramming languages, AMIL for the MCU and ANIMIL for the SPAU, have been defined to facilitate microprogram creation. Translators have been developed for each language that convert the source languages to microcode for the MCU and SPAU. Microcode thus produced can be loaded into the control store of the associated machine.

In addition, simulators for the MCU and SPAU have been implemented to aid microprogram debugging and hardware testing. These simulators accept microcode produced by the translators and provide the user with convenient interactive control of the simulated machine. Both the MCU and SPAU simulators use the same user command language to control simulator execution and the display and modification of the simulated processor's registers.

Two ancillary programs have been written to aid the user in writing microprograms. A pre-processor provides translate-time macro facilities for both AMIL and ANIMIL, and a linkage editor allows programs to be written in modules and combined after translation.

20. ABSTRACT

The SPE is a collection of several interconnected functional components. Two of these components, the Microprogrammed Control Unit (MCU) and the Signal Processing Arithmetic Unit (SPAU), are microprogrammable. This report describes the microprogramming support software developed at NRL to facilitate the development and debugging of microprograms for the Model XB-1 MCU and SPAU.

Microprogramming languages, AMIL for the MCU and ANIMIL for the SPAU, have been defined to facilitate microprogram creation. Translators have been developed for each language that convert the source languages to microcode for the MCU and SPAU. Microcode thus produced can be loaded into the control store of the associated machine.

In addition, simulators for the MCU and SPAU have been implemented to aid microprogram debugging and hardware testing. These simulators accept microcode produced by the translators and provide the user with convenient interactive control of the simulated machine. Both the MCU and SPAU simulators use the same user command language to control simulator execution and the display and modification of the simulated processor's registers.

Two ancillary programs have been written to aid the user in writing microprograms. A preprocessor provides translate-time macro facilities for both AMIL and ANIMIL, and a linkage editor allows programs to be written in modules and combined after translation.

20. ABSTRACT

The SPE is a collection of several interconnected functional components. Two of these components, the Microprogrammed Control Unit (MCU) and the Signal Processing Arithmetic Unit (SPAU), are microprogrammable. This report describes the microprogramming support software developed at NRL to facilitate the development and debugging of microprograms for the Model XB-1 MCU and SPAU.

Microprogramming languages, AMIL for the MCU and ANIMIL for the SPAU, have been defined to facilitate microprogram creation. Translators have been developed for each language that convert the source languages to microcode for the MCU and SPAU. Microcode thus produced can be loaded into the control store of the associated machine.

In addition, simulators for the MCU and SPAU have been implemented to aid microprogram debugging and hardware testing. These simulators accept microcode produced by the translators and provide the user with convenient interactive control of the simulated machine. Both the MCU and SPAU simulators use the same user command language to control simulator execution and the display and modification of the simulated processor's registers.

Two ancillary programs have been written to aid the user in writing microprograms. A preprocessor provides translate-time macro facilities for both AMIL and ANIMIL, and a linkage editor allows programs to be written in modules and combined after translation.

```
                         GROUP 1

## ERROR - INVALID VARIABLE TO BE ACTIVATED OR DEACTIVATED
## ERROR - INVALID FILE FOR INCLUSION
## ERROR - INCLUDE STATEMENT WITHIN INCLUDED TEXT
## ERROR - MISSING OR IMPROPERLY PLACED = IN ASSIGNMENT STATEMENT
## ERROR - VARIABLE IN ASSIGNMENT STATEMENT HAS INVALID LENGTH
## ERROR - MISSING OR IMPROPERLY PLACED IN ASSIGNMENT STATEMENT
## ERROR - MISSING $ AFTER VALUE STRING IN ASSIGNMENT STATEMENT
## ERROR - PREPROCESSOR SYMBOL TABLE FULL

                         GROUP 2

## ERROR - VALUE STRING IN ASSIGNMENT STATEMENT HAS LENGTH GREATER
            THAN 800, FIRST 800 CHARACTERS USED
## ERROR - END OF FILE BEFORE END OF ASSIGNMENT STATEMENT
            STATEMENT IGNORED
## WARNING - CROSS REFERENCE TABLE OVERFLOW
              SUBSEQUENT REFERENCES NOT RECORDED FOR VARIABLE

             ________________________

## ERROR - REPLACED VALUE STRING TOO LONG
            1000 CHARACTERS USED
## WARNING - QUOTE, $ OR AMPERSAND IGNORED IN VARIABLE
```

Fig. 13—Preprocessor error messages

A REGISTER CROSS-REFERENCE LIST FACILITY FOR AMIL PROGRAMS

Description

A register cross-reference list facility can aid the microprogrammer in developing and debugging microprograms because it tells him where machine registers are referenced. The program to be described was developed for AMIL microprograms. The MCU registers searched for in the cross-reference program are the following:

ACSAR
BARA
BARB
BUF
CTR
FSCR
FSDR
FSU
ISR
LSA(0)
LSA(1)
LSA(2)
LSA(3)
LSA(4)
LSA(5)
LSA(6)
LSA(7)
LSA(8)
LSA(9)
LSA(10)
LSA(11)
LSA(12)
LSA(13)
LSA(14)
LSA(15)
LSA(BARA)
LSA(BARB)
LSB(0)
LSB(1)
LSB(2)
LSB(3)
LSB(4)
LSB(5)
LSB(6)
LSB(7)
LSB(8)
LSB(9)
LSB(10)
LSB(11)
LSB(12)
LSB(13)
LSB(14)
LSB(15)
LSB(BARA)
LSB(BARB)
SAR
Z

In keeping with the line orientation of AMIL, references to MCU registers will be listed by line number. Thus for each MCU register name explicitly (not symbolically) referenced in AMIL statements, the cross-reference program will list the name of the register and all the lines wherein AMIL statements explicitly reference it.

Because the ANIMIL translator provides a register cross-reference list (the REGISTER LIST option), there is not a separate program to list registers referenced in ANIMIL microprograms.

Using the Register Cross-Reference Program

The first version of the register cross-reference program has been developed on the KRONOS time-sharing system. To use the program, sign on the system, create and save an AMIL program file, and enter the command

```
EXECUTE
```

to which the system will respond

```
READY.
```

Enter the command

```
ATTACH, DMCUXRE/UN=K5037WU
```

to which the system will respond

```
READY.
```

Enter the command

```
RUN, I=DMCUXRE
```

to initiate program execution. The program will print a heading and then request information by printing the following:

```
ENTER NAME OF AMIL PROGRAM FILE
?
```

Enter the name of the AMIL program file for which a register cross-reference list is desired. The program will then ask

```
SOURCE PROGRAM LIST
?
```

Respond with 'YES' if a listing of the program is desired and 'NO' otherwise. If either of these questions is answered incorrectly, the question will be repeated.

The source program will be listed, if it was requested, with each line preceded by a line number corresponding to the number assigned by the AMIL translator. If the line contains more than 67 characters, the characters in columns 68 through 80 will be printed on the following line. The program will then print the names of registers explicitly referenced in the AMIL program and the line numbers wherein they were referenced. When local storage is addressed by a buffer address register, the cross-reference program will list references to both local storage and the address register.

Figure 14 demonstrates use of the program with the AMIL example microprogram illustrated in Fig. 4.

```
execute
READY.

attach,dmcuxre/un=k5037wu
READY.

run,i=dmcuxre


**** MCU CROSS REFERENCE  VERSION 1.0   74/09/25   16.02.45 ****

ENTER NAME OF AMIL PROGRAM FILE
? addcom

SOURCE PROGRAM LIST
? yes


SOURCE PROGRAM LIST
-------------------

   1
   2
   3
   4
   5
   6 * MCU MICROPROGRAM EXAMPLE
   7 *  ADD COMPLEX NUMBERS
   8
   9 *  ASSUME BSM 0 CONTAINS N PAIRS OF COMPLEX NUMBERS,
  10 *   EACH COMPLEX NUMBER IS STORED IN ONE 32 BIT BSM WORD.
  11 *   SUM SUCCESSIVE PAIRS OF NUMBERS AND STORE
  12 *   THEM IN BSM 1.  ASSUME SUMS DO NOT OVERFLOW.
  13
  14
  15 * INITIALIZE
  16       BARA = 0 $  BUFFER ADDRESS REGISTER A POINTS TO THE
  17                          * INPUT DATA, BUFFER 0 LOCATION 0
  18       BARB = 8192 $  BUFFER ADDRESS REGISTER B POINTS TO THE
  19                          * OUTPUT DATA,  BUFFER 1 LOCATION 0
  20 *     CTR = N-1 , ASSUME CTR IS ONE LESS THAN THE NUMBER OF
  21 *                     PAIRS OF COMPLEX NUMBERS TO BE SUMMED
  22
```

Fig. 14—Use of the cross-reference program with the microprogram from Fig. 4

```
23
24 * FILL UP PIPE
25         INPUT(BUF(BARA),LSA(2)), INC BARA $
26             * LSA(2) = REAL PART OF FIRST NUMBER
27             * INCREMENT POINTER TO INPUT DATA
28
29         INPUT(BUF(BARA),LSA(3)), INC BARA $
30             * LSA(3) = IMAGINARY PART OF FIRST NUMBER
31             * INCREMENT POINTER TO INPUT DATA
32
33         INPUT(BUF(BARA),LSB(2)), INC BARA, SKIP $
34             * LSB(2) = REAL PART OF SECOND NUMBER
35             * INCREMENT POINTER TO INPUT DATA
36             * SKIP NEXT INSTRUCTION SO DON'T OUTPUT GARBAGE
37
38 * LOOP TO SUM NUMBERS AND WRITE THEM OUT
39 .LOOP  OUTPUT(LSB(4),BUF(BARB)),
40         INPUT(BUF(BARA),LSB(2)), INC BARA, INC BARB $
41             * WRITE IMAGINARY PART OF SUM INTO BUFFER
42             * LSB(2) = REAL PART OF SECOND NUMBER OF PAIR
43             * INCREMENT POINTER TO INPUT DATA
44             * INCREMENT POINTER TO OUTPUT DATA
45
46         INPUT(BUF(BARA),LSB(3)), INC BARA,
47         LSA(4) = LSA(2)+LSB(2), IF CTRZERO JUMP TO OUT $
48             * LSB(3) = IMAGINARY PART OF SECOND NUMBER OF PAIR
49             * INCREMENT POINTER TO INPUT DATA
50             * LSA(4) = SUM OF REAL PARTS OF 2 COMPLEX NUMBERS
51             * IF ALL DATA HAVE BEEN READ JUMP OUT OF LOOP
52                     * OTHERWISE DECREMENT COUNTER BY 1
53
54         INPUT(BUF(BARA),LSA(2)),
55         OUTPUT(LSA(4),BUF(BARB)), INC BARA, INC BARB $
56             * LSA(2) = REAL PART OF FIRST NUMBER OF PAIR
57             * WRITE REAL PART OF SUM INTO BUFFER
58             * INCREMENT POINTER TO INPUT DATA
59             * INCREMENT POINTER TO OUTPUT DATA
60
61         LSB(4) = LSA(3)+LSB(3),
62         INPUT(BUF(BARA),LSA(3)), INC BARA, JUMP TO LOOP $
63             * LSB(4) = SUM OF IMAG PARTS OF 2 COMPLEX NUMBERS
64             * LSA(3) = IMAGINARY PART OF FIRST NUMBER OF PAIR
65             * INCREMENT POINTER TO INPUT DATA
66             * CONTINUE LOOPING
67
```

Fig. 14—Use of the cross-reference program with the microprogram from Fig. 4 (Continued)

```
68 * FLUSH PIPE
69 .OUT   OUTPUT(LSA(4),BUF(BARB)), INC BARB $
70             * WRITE REAL PART OF SUM INTO BUFFER
71             * INCREMENT POINTER TO OUTPUT DATA
72
73        LSB(4) = LSA(3)+LSB(3) $
74             * LSB(4) = SUM OF IMAG PARTS OF 2 COMPLEX NUMBERS
75
76        OUTPUT(LSB(4),BUF(BARB)) $
77             * WRITE IMAGINARY PART OF SUM INTO BUFFER
78
79        END $
80
81
82
83
84
85



REGISTER CROSS REFERENCE LIST
-----------------------------

SYMBOL        REFERENCES
------        ----------

BARA           16   25   25   29   29   33   33   40   40   46   46   54
               55   62   62

BARB           18   39   40   55   55   69   69   76

BUF            25   29   33   39   40   46   54   55   62   69   76

CTR            47

LSA(2)         25   47   54

LSA(3)         29   61   62   73

LSA(4)         47   55   69

LSB(2)         33   40   47

LSB(3)         46   61   73

LSB(4)         39   61   73   76



 END.
```

Fig. 14—Use of the cross-reference program with the microprogram from Fig. 4 (Continued)

LINKAGE EDITOR FOR AMIL AND ANIMIL PROGRAMS

Description

A linkage editor combines microprograms that have been separately translated. As discussed earlier, AMIL and ANIMIL microprograms reference other AMIL and ANIMIL microprograms through microinstruction labels and sequencing subcommands and through the entry and external pseudoinstructions. The linkage editor resolves external references, handles code relocation requested by origin pseudoinstructions, and combines the separately translated microprograms into a module amenable to simulation by the MCU or SPAU simulator.

Using the Linkage Editor Program

The first version of the linkage editor has been developed on the KRONOS timesharing system. To use the linkage editor, sign on the system, create and translate several AMIL or ANIMIL microprograms, and enter the command

```
EXECUTE
```

to which the system will respond

```
READY.
```

Enter the command

```
ATTACH,DLKED/UN=K5037WU
```

to which the system will respond

```
READY.
```

Enter the command

```
RUN,I=DLKED
```

to initiate execution of the linkage editor program. The program will pose the following questions:

1. SPAU OR MCU—Answer SPAU or MCU depending on the machine for which the microprograms to be linked were written.

2. ENTER INPUT FILE NAMES AND ADDRESSES—Enter the names of the files that contain object modules to be linked. File names must be separated by commas, and a dollar sign preceded by a blank must terminate the list. The list of file names may extend over multiple input lines. A sample response is

```
P1OBJ, P2OBJ, P3OBJ $
```

The object modules in the list are linked sequentially starting at absolute address 0. To indicate that a module is to be linked to a specific absolute address, write an equal sign and the address after the file name as illustrated in the following example:

```
P1OBJ, P2OBJ = 2048, P3OBJ $
```

In this case, the first module will start at address 0, the second at address 2048, and the third will follow the second.

3. ENTER OBJECT FILE NAME—Enter the name of the file into which the linked object program will be returned.

4. LISTING DESIRED—Answer YES or NO depending on whether a listing of the linked object module is desired.

To illustrate the use of the linkage editor, two sample runs follow (Fig. 15). Also included are the AMIL source programs to clarify linkage editor operation (Fig. 16).

ACKNOWLEDGMENTS

The authors wish to thank those who contributed to the development and testing of the AN/UYK-17 microprogramming support software. In particular, thanks are due to Edmund Freeman, Honey Sue Elovitz, Leonard E. Russo, and Harold H. Smith of the Naval Research Laboratory and James L. Schilling of the Naval Undersea Center, San Diego, California.

REFERENCES

1. W. R. Smith, J. P. Ihnat, H. H. Smith, N. M. Head, Jr., E. Freeman, Y. S. Wu, and B. Wald, "AN/UYK-17 (XB-1) (V) Signal Processing Element Architecture," NRL Report 7704, June 7, 1974.
2. T. G. Rauscher and A. K. Agrawala, "On the Syntax and Semantics of Horizontal Microprogramming Languages," in *Proceedings of the ACM National Conference*, Association for Computing Machinery, New York, 1973.
3. T. G. Rauscher, "The Formal Syntax and Semantics of the ANIMIL Microprogramming Language," NRL Report, in preparation.

```
execute
READY.

attach,dlked/un=k5037wu
READY.

run,i=dlked

 SPAU OR MCU
? mcu
 ENTER INPUT FILE NAMES AND ADDRESSES
? plx,p2x,p3x
?  $
 ENTER OUTPUT FILE NAME
? px
 LISTING DESIRED
? yes


 NEW ORIGIN TABLE
   ORIG    CNT
-------------
      0      2
     43      4
   1047      4
   1051      4
   1076      2


 NEW ENTRY POINTS
 ENTRY NAME    ADDRESS
---------------------
 FIN              45
 MOD1           1047
 X              1049
 MOD2           1051
```

Fig. 15—Two samples of linkage editor usage

```
  NEW  OLD                                     FIELDS(1-17)
 ADDR ADDR LINE    1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16      17
--------------------------------------------------------------------------
    0    0    9    0  0  0  0  0  3  0  0  0  0  0  8  0  0  1  0      11
    1    1   10    0  4  0  0  0  0  0  0  0  0  0  0  0  0  0  0      43
   43   43   14    0  0  0  0  0  3  0  0  0  0  0  8  0  0  2  0      22
   44   44   15    0  4  0  0  0  0  0  0  0  0  0  0  0  0  0  0    1049
   45   45   17    0  0  0  0  0  3  0  0  0  0  0  8  0  0  3  0      33
   46   46   18    0  0  0  0  0  0  0  0  0  0  0  3  0  0  0  0       0


 1047 1000   10    0  0  0  0  0  3  0  0  0  0  0  8  0  0  5  0      55
 1048 1001   11    0  4  0  0  0  0  0  0  0  0  0  0  0  0  0  0      45
 1049 1002   13    0  0  0  0  0  3  0  0  0  0  0  8  0  0  6  0      66
 1050 1003   14    0  4  0  0  0  0  0  0  0  0  0  0  0  0  0  0    1051


 1051    0    7    0  0  0  0  0  3  0  0  0  0  0  8  0  0  7  0      77
 1052    1    8    0  0  0  2  0  0  0  0  0  0  0  0  0  0  0  0       0
 1053    2    9    0  0  0  0  2  0  0  0  0  0  0  0  0  0  0  0       0
 1054    3   10    0  4  0  0  0  0  0  0  0  0  0  0  0  0  0  0    1076
 1076   25   14    0  0  0  0  0  3  0  0  0  0  0  8  0  0  8  0      88
 1077   26   15    0  4  0  0  0  0  0  0  0  0  0  0  0  0  0  0    1047


 COLLAPSED ORIGIN TABLE
   ORIG   CNT
--------------
      0     2
     43     4
   1047     8
   1076     2


 END.

SS        0.646 SECS.

RUN COMPLETE.
```

Fig. 15—Two samples of linkage editor usage (Continued)

```
execute
READY.

attach,dlked/un=k5037wu
READY.

run,i=dlked

 SPAU OR MCU
? mcu
 ENTER INPUT FILE NAMES AND ADDRESSES
? p1x,p2x=2000,p3x $
 ENTER OUTPUT FILE NAME
? py
 LISTING DESIRED
? yes


 NEW ORIGIN TABLE
   ORIG   CNT
-------------
      0      2
     43      4
   3000      4
   3004      4
   3029      2


 NEW ENTRY POINTS
 ENTRY NAME   ADDRESS
---------------------
 FIN             45
 MOD1          3000
 X             3002
 MOD2          3004
```

Fig. 15—Two samples of linkage editor usage (Continued)

```
  NEW   OLD                                       FIELDS(1-17)
 ADDR ADDR  LINE     1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16       17
-------------------------------------------------------------------------------
    0     0     9    0  0  0  0  0  3  0  0  0  0  0  8  0  0  1  0       11
    1     1    10    0  4  0  0  0  0  0  0  0  0  0  0  0  0  0  0       43
   43    43    14    0  0  0  0  0  3  0  0  0  0  0  8  0  0  2  0       22
   44    44    15    0  4  0  0  0  0  0  0  0  0  0  0  0  0  0  0     3002
   45    45    17    0  0  0  0  0  3  0  0  0  0  0  8  0  0  3  0       33
   46    46    18    0  0  0  0  0  0  0  0  0  0  0  3  0  0  0  0        0

 3000  1000    10    0  0  0  0  0  3  0  0  0  0  0  8  0  0  5  0       55
 3001  1001    11    0  4  0  0  0  0  0  0  0  0  0  0  0  0  0  0       45
 3002  1002    13    0  0  0  0  0  3  0  0  0  0  0  8  0  0  6  0       66
 3003  1003    14    0  4  0  0  0  0  0  0  0  0  0  0  0  0  0  0     3004

 3004     0     7    0  0  0  0  0  3  0  0  0  0  0  8  0  0  7  0       77
 3005     1     8    0  0  0  2  0  0  0  0  0  0  0  0  0  0  0  0        0
 3006     2     9    0  0  0  0  2  0  0  0  0  0  0  0  0  0  0  0        0
 3007     3    10    0  4  0  0  0  0  0  0  0  0  0  0  0  0  0  0     3029
 3029    25    14    0  0  0  0  0  3  0  0  0  0  0  8  0  0  8  0       88
 3030    26    15    0  4  0  0  0  0  0  0  0  0  0  0  0  0  0  0     3000


 COLLAPSED ORIGIN TABLE
   ORIG    CNT
-------------
      0      2
     43      4
   3000      8
   3029      2


 END.

SS        0.623 SECS.

RUN COMPLETE.
```

Fig. 15—Two samples of linkage editor usage (Continued)

```
execute
READY.

attach,damil/un=k5037wu
READY.

run,i=damil

            AMIL TRANSLATOR      VERSION 1.0.256.73

 SOURCE AND OBJECT FILE NAMES
? pl,plx
 LISTINGS -SOURCE,OBJECT,STATS,SYM TAB,COND LIST KEY
? yes,yes,yes,yes,yes


 SOURCE LISTING

 ADDR  LINE          AMIL STATEMENT
----------------------------------------------------------------

    0      1
    0      2 * AMIL PROGRAM SEGMENT #1 TO DEMONSTRATE LINK EDIT FACILITIE
               S
    0      3
    0      4
    0      5        EXT X $
    0      6
    0      7        ENT FIN $
    0      8
    0      9        LSA(1) = 11 $
    1     10        JUMP TO START $
    2     11
    2     12        ORIG 43 $
   43     13
   43     14 .START LSA(2) = 22 $
   44     15        JUMP TO X $
   45     16
   45     17 .FIN   LSA(3) = 33 $
   46     18        Z = 0 $
   47     19
   47     20        END $


  ORIGIN TABLE
   ORIG   CNT
-------------
     0      2
    43      4
```

Fig. 16—AMIL source programs for Fig. 15

```
   SYMBOL TABLE                                  CROSS REFERENCE LIST
 SYMBOL         VALUE                          LINE NUMBER OF OCCURENCE
-------------------------------------------------------------------------------
 FIN              45          7    17
 START            43         10    14


 EXTERNALS
NAME                                           LINE NUMBER OF OCCURENCE
-------------------------------------------------------------------------------
 X                            5    15


 ENTRY POINT TABLE
  NAME    INSTR.NO.
--------------------
 FIN             5


 OBJECT LISTING
                                FIELDS(1-17)
 ADDR. LINE    1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16      17
---------------------------------------------------------------------------
    0     9    0   0   0   0   0   3   0   0   0   0   0   8   0   0   1   0      11
    1    10    0   4   0   0   0   0   0   0   0   0   0   0   0   0   0   0      43


   43    14    0   0   0   0   0   3   0   0   0   0   0   8   0   0   2   0      22
   44    15    0   4   0   0   0   0   0   0   0   0   0   0   0   0   0   0   99999
   45    17    0   0   0   0   0   3   0   0   0   0   0   8   0   0   3   0      33
   46    18    0   0   0   0   0   0   0   0   0   0   0   3   0   0   0   0       0

 NO COMPILATION ERRORS WERE ENCOUNTERED


 FIELD UTILIZATION STATISTICS IN PER CENT
   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17
---------------------------------------------------------------------------
   0  34   0   0   0  50   0   0   0   0  67  67   0   0  50   0  84

 COMPILATION TIME =     .74 SECONDS
 END.


SS        1.111 SECS.

RUN COMPLETE.
```

Fig. 16—AMIL source programs for Fig. 15 (Continued)

```
execute
READY.

attach,damil/un=k5037wu
READY.

run,i=damil

           AMIL TRANSLATOR      VERSION 1.0.256.73

 SOURCE AND OBJECT FILE NAMES
? p2,p2x
 LISTINGS -SOURCE,OBJECT,STATS,SYM TAB,COND LIST KEY
? yes,yes,yes,yes,yes


 SOURCE LISTING

 ADDR  LINE           AMIL STATEMENT
-----------------------------------------------------------------------

    0      1
    0      2 * AMIL PROGRAM SEGMENT #2 TO DEMONSTRATE LINK EDIT FACILITIE
               S
    0      3
    0      4        EXT FIN,MOD2 $
    0      5
    0      6        ENT MOD1,X $
    0      7
    0      8        ORIG 1000 $
 1000      9
 1000     10 .MOD1  LSA(5) = 55 $
 1001     11        JUMP TO FIN $
 1002     12
 1002     13 .X     LSA(6) = 66 $
 1003     14        JUMP TO MOD2 $
 1004     15
 1004     16        END $
```

Fig. 16—AMIL source programs for Fig. 15 (Continued)

```
  ORIGIN TABLE
   ORIG   CNT
-------------
   1000    4
   SYMBOL TABLE                        CROSS REFERENCE LIST
 SYMBOL         VALUE                 LINE NUMBER OF OCCURENCE
----------------------------------------------------------------------------
 MOD1            1000        6   10
 X               1002        6   13


 EXTERNALS
NAME                                  LINE NUMBER OF OCCURENCE
----------------------------------------------------------------------------
 FIN                         4   11
 MOD2                        4   14


 ENTRY POINT TABLE
  NAME    INSTR.NO.
-------------------
 MOD1           1
 X              3


 OBJECT LISTING
                              FIELDS(1-17)
 ADDR. LINE    1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16      17
----------------------------------------------------------------------


 1000   10     0  0  0  0  0  3  0  0  0  0  0  8  0  0  5  0      55
 1001   11     0  4  0  0  0  0  0  0  0  0  0  0  0  0  0  0   99999
 1002   13     0  0  0  0  0  3  0  0  0  0  0  8  0  0  6  0      66
 1003   14     0  4  0  0  0  0  0  0  0  0  0  0  0  0  0  0   99999

 NO COMPILATION ERRORS WERE ENCOUNTERED


 FIELD UTILIZATION STATISTICS IN PER CENT
   1   2   3   4   5   6   7   8   9   10   11   12   13   14   15   16  17
----------------------------------------------------------------------
   0  50   0   0   0  50   0   0   0   0   50   50   0    0   50    0 100

 COMPILATION TIME =      .61 SECONDS
 END.


SS       0.921 SECS.

RUN COMPLETE.
```

Fig. 16—AMIL source programs for Fig. 15 (Continued)

```
execute
READY.

attach,damil/un=k5037wu
READY.

run,i=damil

           AMIL TRANSLATOR      VERSION 1.0.256.73

 SOURCE AND OBJECT FILE NAMES
? p3,p3x
 LISTINGS -SOURCE,OBJECT,STATS,SYM TAB,COND LIST KEY
? yes,qes,qes,yes,yes


 SOURCE LISTING

 ADDR  LINE          AMIL STATEMENT
-------------------------------------------------------------------

   0      1
   0      2 * AMIL PROGRAM SEGMENT #3 TO DEMONSTRATE LINK EDIT FACILITIE
            S
   0      3
   0      4      EXT MOD1 $
   0      5      ENT MOD2 $
   0      6
   0      7 .MOD2 LSA(7) = 77 $
   1      8      INC BARA $
   2      9      INC BARB $
   3     10      JUMP TO NEXT $
   4     11
   4     12      ORIG 25 $  NOTE THAT ORIGIN IS RELATIVE TO START OF PR
            OGRAM SEGMENT
  25     13
  25     14 .NEXT LSA(8) = 88 $
  26     15      JUMP TO MOD1 $
  27     16
  27     17      END $
```

Fig. 16—AMIL source programs for Fig. 15 (Continued)

```
  ORIGIN TABLE
   ORIG   CNT
-------------
      0     4
     25     2

   SYMBOL TABLE                            CROSS REFERENCE LIST
 SYMBOL          VALUE                   LINE NUMBER OF OCCURENCE
-------------------------------------------------------------------------
 MOD2                0          5    7
 NEXT               25         10   14


 EXTERNALS
NAME                                     LINE NUMBER OF OCCURENCE
-------------------------------------------------------------------------
 MOD1                           4   15


 ENTRY POINT TABLE
  NAME    INSTR.NO.
-------------------
 MOD2             1


 OBJECT LISTING
                                    FIELDS(1-17)
 ADDR. LINE    1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16      17
---------------------------------------------------------------------
     0    7    0  0  0  0  0  3  0  0  0  0  0  8  0  0  7  0      77
     1    8    0  0  0  2  0  0  0  0  0  0  0  0  0  0  0  0       0
     2    9    0  0  0  0  2  0  0  0  0  0  0  0  0  0  0  0       0
     3   10    0  4  0  0  0  0  0  0  0  0  0  0  0  0  0  0      25


    25   14    0  0  0  0  0  3  0  0  0  0  0  8  0  0  8  0      88
    26   15    0  4  0  0  0  0  0  0  0  0  0  0  0  0  0  0   99999

 NO COMPILATION ERRORS WERE ENCOUNTERED


 FIELD UTILIZATION STATISTICS IN PER CENT
   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17
---------------------------------------------------------------------
   0  34   0  17  17  34   0   0   0   0  34  34   0   0  34   0  67

 COMPILATION TIME =     .68 SECONDS
 END.


SS       1.053 SECS.

RUN COMPLETE.
```

Fig. 16—AMIL source programs for Fig. 15 (Continued)

Appendix A
MCU CONTROL FIELDS

The definitions of the individual MCU control fields, listed below, are summarized in Table A1.

	Field Width	Field
1.	4	Condition
2.	3	Next Address
3.	3	Auxiliary Transfer
4.	2	BARA Source
5.	2	BARB Source
6.	2	LSA Source
7.	2	LSB Source
8.	3	Buffer Memory Write Operation
9.	1	FSCR Control
10.	1	JSR Control
11.	4	Adder Inputs
12.	4	Adder Operation
13.	4	LSA Read Address
14.	4	LSA Write Address
15.	4	LSB Read Address
16.	4	LSB Write Address
17.	16	Literal

20. ABSTRACT

The SPE is a collection of several interconnected functional components. Two of these components, the Microprogrammed Control Unit (MCU) and the Signal Processing Arithmetic Unit (SPAU), are microprogrammable. This report describes the microprogramming support software developed at NRL to facilitate the development and debugging of microprograms for the Model XB-1 MCU and SPAU.

Microprogramming languages, AMIL for the MCU and ANIMIL for the SPAU, have been defined to facilitate microprogram creation. Translators have been developed for each language that convert the source languages to microcode for the MCU and SPAU. Microcode thus produced can be loaded into the control store of the associated machine.

In addition, simulators for the MCU and SPAU have been implemented to aid microprogram debugging and hardware testing. These simulators accept microcode produced by the translators and provide the user with convenient interactive control of the simulated machine. Both the MCU and SPAU simulators use the same user command language to control simulator execution and the display and modification of the simulated processor's registers.

Two ancillary programs have been written to aid the user in writing microprograms. A preprocessor provides translate-time macro facilities for both AMIL and ANIMIL, and a linkage editor allows programs to be written in modules and combined after translation.

20. ABSTRACT

The SPE is a collection of several interconnected functional components. Two of these components, the Microprogrammed Control Unit (MCU) and the Signal Processing Arithmetic Unit (SPAU), are microprogrammable. This report describes the microprogramming support software developed at NRL to facilitate the development and debugging of microprograms for the Model XB-1 MCU and SPAU.

Microprogramming languages, AMIL for the MCU and ANIMIL for the SPAU, have been defined to facilitate microprogram creation. Translators have been developed for each language that convert the source languages to microcode for the MCU and SPAU. Microcode thus produced can be loaded into the control store of the associated machine.

In addition, simulators for the MCU and SPAU have been implemented to aid microprogram debugging and hardware testing. These simulators accept microcode produced by the translators and provide the user with convenient interactive control of the simulated machine. Both the MCU and SPAU simulators use the same user command language to control simulator execution and the display and modification of the simulated processor's registers.

Two ancillary programs have been written to aid the user in writing microprograms. A preprocessor provides translate-time macro facilities for both AMIL and ANIMIL, and a linkage editor allows programs to be written in modules and combined after translation.

20. ABSTRACT

The SPE is a collection of several interconnected functional components. Two of these components, the Microprogrammed Control Unit (MCU) and the Signal Processing Arithmetic Unit (SPAU), are microprogrammable. This report describes the microprogramming support software developed at NRL to facilitate the development and debugging of microprograms for the Model XB-1 MCU and SPAU.

Microprogramming languages, AMIL for the MCU and ANIMIL for the SPAU, have been defined to facilitate microprogram creation. Translators have been developed for each language that convert the source languages to microcode for the MCU and SPAU. Microcode thus produced can be loaded into the control store of the associated machine.

In addition, simulators for the MCU and SPAU have been implemented to aid microprogram debugging and hardware testing. These simulators accept microcode produced by the translators and provide the user with convenient interactive control of the simulated machine. Both the MCU and SPAU simulators use the same user command language to control simulator execution and the display and modification of the simulated processor's registers.

Two ancillary programs have been written to aid the user in writing microprograms. A preprocessor provides translate-time macro facilities for both AMIL and ANIMIL, and a linkage editor allows programs to be written in modules and combined after translation.

20. ABSTRACT

The SPE is a collection of several interconnected functional components. Two of these components, the Microprogrammed Control Unit (MCU) and the Signal Processing Arithmetic Unit (SPAU), are microprogrammable. This report describes the microprogramming support software developed at NRL to facilitate the development and debugging of microprograms for the Model XB-1 MCU and SPAU.

Microprogramming languages, AMIL for the MCU and ANIMIL for the SPAU, have been defined to facilitate microprogram creation. Translators have been developed for each language that convert the source languages to microcode for the MCU and SPAU. Microcode thus produced can be loaded into the control store of the associated machine.

In addition, simulators for the MCU and SPAU have been implemented to aid microprogram debugging and hardware testing. These simulators accept microcode produced by the translators and provide the user with convenient interactive control of the simulated machine. Both the MCU and SPAU simulators use the same user command language to control simulator execution and the display and modification of the simulated processor's registers.

Two ancillary programs have been written to aid the user in writing microprograms. A preprocessor provides translate-time macro facilities for both AMIL and ANIMIL, and a linkage editor allows programs to be written in modules and combined after translation.

Appendix B
SPAU CONTROL FIELDS

The definitions of the individual SPAU control fields, listed below, are summarized in Table B1 and detailed in the text that follows.

Summary of Control Word Fields

	Field Width	Field
1.	5	X Read Address
2.	4	X Write Address
3.	4	Y Read Address
4.	4	Y Write Address
5.	5	W Address
6.	16	Literal
7.	3	W Source
8.	2	Z Register Source
9.	3	X1 Source
10.	3	X2 Source
11.	3	Y1 Source
12.	3	Y2 Source
13.	3	Literal Destination
14.	1	CFS Write
15.	2	W Store Shift Control
16.	2	Counter Decrement
17.	2	Channel Control
18.	3	Counter Input
19.	3	Channel A Source
20	3	Channel B Source
21.	2	Adder 5 Destination
22.	2	Adder 6 Destination
23.	2	Adder 7 Destination
24.	2	Local Store Remote Addressing
25.	1	Buffer Pointer Hold
26.	2	Multiplier Outputs
27.	1	I/O Interrupt
28.	4	Condition
29.	1	Condition Mode
30.	3	Control Sequence
31.	2	Adder 1 Operation
32.	2	Adder 2 Operation
33.	2	Adder 3 Operation
34.	2	Adder 4 Operation

	Field Width	Field
35.	2	Adder 5 Operation
36.	2	Adder 6 Operation
37.	2	Adder 7 Operation
38.	2	R1 Source
39.	2	R2 Source
40.	2	R3 Source
41.	2	R4 Source
42.	2	A1 Left Source
43.	2	A1 Right Source
44.	2	A2 Left Source
45.	2	A2 Right Source
46.	2	A3 Left Source
47.	2	A3 Right Source
48.	2	A4 Left Source
49.	2	A4 Right Source
50.	2	A5 Left Source
51.	2	A5 Right Source
52.	2	A6 Left Source
53.	2	A6 Right Source
54.	2	A7 Left Source
55.	2	A7 Right Source
56.	2	M1 Left Source
57.	2	M1 Right Source
58.	2	M2 Left Source
59.	2	M2 Right Source
60.	2	M3 Left Source
61.	2	M3 Right Source
62.	2	M4 Left Source
63.	2	M4 Right Source

CONTROL FIELD DEFINITIONS

Field 1. The X Read Address field specifies the location in the X1 or X2 store that is to be a data source, unless code point 1 of field 24 is specified. Code points 16 to 31 are unused.

Code Point	Store Address
0	0
.	.
.	.
.	.
15	15

20. ABSTRACT

The SPE is a collection of several interconnected functional components. Two of these components, the Microprogrammed Control Unit (MCU) and the Signal Processing Arithmetic Unit (SPAU), are microprogrammable. This report describes the microprogramming support software developed at NRL to facilitate the development and debugging of microprograms for the Model XB-1 MCU and SPAU.

Microprogramming languages, AMIL for the MCU and ANIMIL for the SPAU, have been defined to facilitate microprogram creation. Translators have been developed for each language that convert the source languages to microcode for the MCU and SPAU. Microcode thus produced can be loaded into the control store of the associated machine.

In addition, simulators for the MCU and SPAU have been implemented to aid microprogram debugging and hardware testing. These simulators accept microcode produced by the translators and provide the user with convenient interactive control of the simulated machine. Both the MCU and SPAU simulators use the same user command language to control simulator execution and the display and modification of the simulated processor's registers.

Two ancillary programs have been written to aid the user in writing microprograms. A preprocessor provides translate-time macro facilities for both AMIL and ANIMIL, and a linkage editor allows programs to be written in modules and combined after translation.

20. ABSTRACT

The SPE is a collection of several interconnected functional components. Two of these components, the Microprogrammed Control Unit (MCU) and the Signal Processing Arithmetic Unit (SPAU), are microprogrammable. This report describes the microprogramming support software developed at NRL to facilitate the development and debugging of microprograms for the Model XB-1 MCU and SPAU.

Microprogramming languages, AMIL for the MCU and ANIMIL for the SPAU, have been defined to facilitate microprogram creation. Translators have been developed for each language that convert the source languages to microcode for the MCU and SPAU. Microcode thus produced can be loaded into the control store of the associated machine.

In addition, simulators for the MCU and SPAU have been implemented to aid microprogram debugging and hardware testing. These simulators accept microcode produced by the translators and provide the user with convenient interactive control of the simulated machine. Both the MCU and SPAU simulators use the same user command language to control simulator execution and the display and modification of the simulated processor's registers.

Two ancillary programs have been written to aid the user in writing microprograms. A preprocessor provides translate-time macro facilities for both AMIL and ANIMIL, and a linkage editor allows programs to be written in modules and combined after translation.

20. ABSTRACT

The SPE is a collection of several interconnected functional components. Two of these components, the Microprogrammed Control Unit (MCU) and the Signal Processing Arithmetic Unit (SPAU), are microprogrammable. This report describes the microprogramming support software developed at NRL to facilitate the development and debugging of microprograms for the Model XB-1 MCU and SPAU.

Microprogramming languages, AMIL for the MCU and ANIMIL for the SPAU, have been defined to facilitate microprogram creation. Translators have been developed for each language that convert the source languages to microcode for the MCU and SPAU. Microcode thus produced can be loaded into the control store of the associated machine.

In addition, simulators for the MCU and SPAU have been implemented to aid microprogram debugging and hardware testing. These simulators accept microcode produced by the translators and provide the user with convenient interactive control of the simulated machine. Both the MCU and SPAU simulators use the same user command language to control simulator execution and the display and modification of the simulated processor's registers.

Two ancillary programs have been written to aid the user in writing microprograms. A preprocessor provides translate-time macro facilities for both AMIL and ANIMIL, and a linkage editor allows programs to be written in modules and combined after translation.

Field 2. The X Write Address field specifies the location in the X1 or X2 store that is to be a data destination, unless code point 1 of field 24 is specified.

Code Point	Store Address
0	0
.	.
.	.
.	.
15	15

Field 3. The Y Read Address field specifies the location in the Y1 or Y2 store that is to be a data source, unless code point 2 of field 24 is specified.

Code Point	Store Address
0	0
.	.
.	.
.	.
15	15

Field 4. The Y Write Address field specifies the location in the Y1 or Y2 store that is to be a data destination, unless code point 2 of field 24 is specified.

Code Point	Store Address
0	0
.	.
.	.
.	.
15	15

Field 5. The W Address field specifies the location in the W store that is to be a data source and/or destination.

Code Point	Store Address
0	0
.	.
.	.
.	.
31	31

Field 6. The Literal field defines a 16-bit value that may be used as a source of data and/or addresses to various portions of the SPAU arithmetic section and address generator.

Field 7. The W Source field specifies the data source for the W store.

Code Point	Source
0	None
1	X1
2	X2
3	CTRJ
4	CTRK
5	A5
6	A6
7	A7

Field 8. The Z-Register Source field specifies the data source for the Z-registers (Z1, Z2, Z3, and Z4).

Code Point	Source
0	None
1	CFS
2	Load Z1 and Z2 with X1 and X2
3	Load Z3 and Z4 with Y1 and Y2

Field 9. The X1 Source field specifies the data source for the X1 half of the X store.

Code Point	Source
0	None or Literal (see Field 13)
1	A1
2	A2
3	A3
4	A4
5	Upper half of Channel A (MSB)
6	Upper half of Channel B (MSB)
7	W

Field 10. The X2 Source field specifies the data source for the X2 half of the X store.

Code Point	Source
0	None or Literal (see Field 13)
1	A1
2	A2
3	A3
4	A4
5	Lower half of Channel A (LSB)
6	Lower half of Channel B (LSB)
7	W

Field 11. The Y1 Source field specifies the data source for the Y1 half of the Y store.

Code Point	Source
0	None or Literal (see Field 13)
1	A1
2	A2
3	A3
4	A4
5	Upper half of Channel A (MSB)
6	Upper half of Channel B (MSB)
7	W

Field 12. The Y2 Source field specifies the data source for the Y2 half of the Y store.

Code Point	Source
0	None or Literal (see Field 13)
1	A1
2	A2
3	A3
4	A4
5	Lower half of Channel A (LSB)
6	Lower half of Channel B (LSB)
7	W

Field 13. The Literal Destination field specifies the destination of the value in the literal field of the command word.

Code Point	Destination
0	None
1	X1
2	X2
3	Y1
4	Y2
5	Both R1 and R2
6	Both R3 and R4
7	ACSAR

Field 14. The CFS Write field controls the CFS write operation. The data source is registers Z1, Z2, Z3, and Z4; data transfer simultaneously into the four 16-bit fields of the 64-bit CFS

Code Point	Operation
0	None (no write)
1	Z1, Z2, Z3, Z4 $\rightarrow$ CFS

Field 15. The W Store Shift Control field specifies a shift to be applied to data being read out of the W store.

Code Point	Amount of Shift
0	None
1	Right one bit
2	Left one bit
3	Unused

Field 16. The Counter Decrement Control field specifies a counter decrement of 1.

Code Point	Counter Decrement
0	None
1	CTRI
2	CTRJ
3	CTRK

Field 17. The Channel Control field specifies a change in Channel A and B data line connections.

Code Point	Operation
0	None (no change)
1	Connect A and B for 1-bit right shift on input data
2	Resume normal, unshifted connection
3	Exchange A and B buffer pointers

Field 18. The Counter Set field specifies a counter register write operation.

Code Point	Operation
0	None
1	W → CTRI
2	W → CTRJ
3	W → CTRK
4	Literal → CTRI
5	Literal → CTRJ
6	Literal → CTRK
7	Unused

Field 19. The Channel A Source field specifies the sources for a 32-bit data output transfer over Channel A.

Code Point	Source
0	None (no operation)
1	R1 and R2
2	R1 and R3
3	R2 and R4
4	R3 and R4
5	X
6	Y
7	Unused

Field 20. The Channel B Source field specifies the sources for a 32-bit data output transfer over Channel B.

Code Point	Source
0	None (no operation)
1	R1 and R2
2	R1 and R3
3	R2 and R4
4	R3 and R4
5	X
6	Y
7	Unused

Field 21. The Adder 5 Destination field specifies the destination of the results of the Adder 5 operation.

Code Point	Destination
0	None
1	BARA
2	INCA
3	BARA and INCA

Field 22. The Adder 6 Destination field specifies the destination of the results of the Adder 6 operation.

Code Point	Destination
0	None
1	BARB
2	INCB
3	BARB and INCB

Field 23. The Adder 7 Destination field specifies the destination of the results of the Adder 7 operation.

Code Point	Destination
0	None
1	RAR
2	INCR
3	R7

Field 24. The Local Store Remote Address (LSRA) field specifies the right-most four or five bits of the RAR as the address source for reads and writes of stores X, Y, or W.

Code Point	Operation
0	None (address from Literal address fields)
1	X(RAR)
2	Y(RAR)
3	W(RAR)

Field 25. The Buffer Pointer Hold field specifies that, during a write into BARA and/or BARB, the higher four bits of both registers shall remain unchanged.

Code Point	Definition
0	Full BARA, BARB write
1	Hold high bits of BARA, BARB

Field 26. The Multiplier Outputs field specifies the 16 bits to be selected as the output products of all four multipliers.

Code Point	Product Output
0	Bits 30 to 15 (most significant)
1	Bits 29 to 14
2	Bits 27 to 12
3	Bits 15 to 0 (least significant)

Field 27. The I/O Interrupt field specifies activation of the SPAU interrupt condition.

Code Point	Definition
0	No operation
1	Raise the I/O interrupt line and halt

Field 28. The Condition field specifies one of 16 conditions to be used as a basis for determining the next control store address sequence.

Code Point	Condition
0	Unconditional
1	CTRI zero
2	CTRJ zero
3	CTRK zero
4	AS adder overflow
5	AS adder magnitude limit
6	Adder 1 MSB
7	Adder 2 MSB
8	Adder 3 MSB
9	Adder 4 MSB
10	Adder 1 zero
11	Adder 2 zero
12	Adder 3 zero
13	Adder 4 zero
14	R7 MSB
15	R7 zero

The AS adder overflow condition is true when any of the AS adders overflows in two's complement arithmetic (the two adder inputs have the same sign and the adder output has the opposite sign). The AS adder magnitude limit condition is true when any one of the AS adder outputs is greater than or equal to 1/2 or is less than –1/2 in two's complement fractional representation (the two high-order bits of any AS adder output have opposite values).

Field 29. The Condition Mode field specifies a true or false test on the condition specified in the Condition field.

Code Point	Definition
0	True
1	False

Field 30. The Control Sequence field specifies the next control store address sequence.

Code Point	Operation
0	STEP
1	SKIP
2	SAVE
3	CALL
4	JUMP TO LITERAL value as an address
5	JUMP TO ACSAR
6	JUMP TO W
7	JUMP TO R7

Field 31. The Adder 1 Operation field specifies the arithmetic or logical operation performed by Adder 1.

Code Point	Operation
0	Left input – right input + 0
1	Left input + right input + 0
2	Left input + right input + 1
3	Logical AND

Field 32. The Adder 2 Operation field specifies the arithmetic or logical operation performed by Adder 2.

Code Point	Operation
0	Left input – right input + 0
1	Left input + right input + 0
2	Left input + right input + 1
3	Logical AND

Field 33. The Adder 3 Operation field specifies the arithmetic or logical operation performed by Adder 3.

Code Point	Operation
0	Left input – right input + 0
1	Left input + right input + 0
2	Left input + right input + 1
3	Logical AND

Field 34. The Adder 4 Operation field specifies the arithmetic or logical operation performed by Adder 4.

Code Point	Operation
0	Left input – right input + 0
1	Left input + right input + 0
2	Left input + right input + 1
3	Logical AND

Field 35. The Adder 5 Operation field specifies the arithmetic or logical operation performed by Adder 5

Code Point	Operation
0	Left input + right input + 0
1	Left input + right input + 1
2	Left input – right input + 0
3	0 + right input

Field 36. The Adder 6 Operation field specifies the arithmetic or logical operation performed by Adder 6.

Code Point	Operation
0	Left input + right input + 0
1	Left input + right input + 1
2	Left input – right input + 0
3	0 + right input

Field 37. The Adder 7 Operation field specifies the arithmetic or logical operation performed by Adder 7.

Code Point	Operation
0	Left input + right input + 0
1	Left input + right input + 1
2	Left input – right input + 0
3	Left input + 0

Field 38. The R1 Source field specifies the data source for R1.

Code Point	Source
0	None or Literal
1	A1
2	A2
3	M1

Field 39. The R2 Source field specifies the data source for R2.

Code Point	Source
0	None or Literal
1	A1
2	A2
3	M2

Field 40. The R3 Source field specifies the data source for R3.

Code Point	Source
0	None or Literal
1	A3
2	A4
3	M3

Field 41. The R4 Source field specifies the data source for R4.

Code Point	Source
0	None or Literal
1	A3
2	A4
3	M4

Field 42. The A1 Left Source field specifies the source for the left input of Adder 1.

Code Point	Source
0	P1
1	X1
2	Y1
3	R1

Field 43. The A1 Right Source field specifies the source for the right input of Adder 1.

Code Point	Source
0	P2
1	Y1
2	R1
3	Unused

Field 44. The A2 Left Source field specifies the source for the left input of Adder 2.

Code Point	Source
0	Y1
1	X2
2	Y2
3	R2

Field 45. The A2 Right Source field specifies the source for the right input of Adder 2.

Code Point	Source
0	A1
1	P1
2	P2
3	Y2

Field 46. The A3 Left Source field specifies the source for the left input of Adder 3.

Code Point	Source
0	P3
1	Y1
2	Y2
3	R3

Field 47. The A3 Right Source field specifies the source for the right input of Adder 3.

Code Point	Source
0	P4
1	R3
2	Y2
3	X2

Field 48. The A4 Left Source field specifies the source for the left input of Adder 4.

Code Point	Source
0	Y2
1	Y1
2	R4
3	Unused

Field 49. The A4 Right Source field specifies the source for the right input of Adder 4.

Code Point	Source
0	A3
1	X1
2	P3
3	P4

Field 50. The A5 Left Source field specifies the source for the left input of Adder 5. Code points 2 and 3 are unused.

Code Point	Source
0	BARA
1	Literal

Field 51. The A5 Right Source field specifies the source for the right input of Adder 5. Code points 2 and 3 are unused.

Code Point	Source
0	INCA
1	W

Field 52. The A6 Left Source field specifies the source for the left input of Adder 6. Code points 2 and 3 are unused.

Code Point	Source
0	BARB
1	Literal

Field 53. The A6 Right Source field specifies the source for the right input of Adder 6. Code points 2 and 3 are unused.

Code Point	Source
0	INCB
1	W

Field 54. The A7 Left Source field specifies the source for the left input of Adder 7. Code points 2 and 3 are unused.

Code Point	Source
0	RAR
1	Literal

Field 55. The A7 Right Source field specifies the source for the right input of Adder 7. Code points 2 and 3 are unused.

Code Point	Source
0	INCR
1	W

Field 56. The M1 Left Source field specifies the source for the left input of Multiplier 1.

Code Point	Source
0	X2
1	X1
2	Y1
3	Y2

Field 57. The M1 Right Source field specifies the source for the right input of Multiplier 1.

Code Point	Source
0	Z1
1	X1
2	Y1
3	R1

Field 58. The M2 Left Source field specifies the source for the left input of Multiplier 2.

Code Point	Source
0	X1
1	X2
2	Y2
3	Y1

Field 59. The M2 Right Source field specifies the source for the right input of Multiplier 2.

Code Point	Source
0	Z2
1	Y2
2	X2
3	R2

Field 60. The M3 Left Source field specifies the source for the left input of Multiplier 3.

Code Point	Source
0	X2
1	X1
2	Y1
3	Y2

Field 61. The M3 Right Source field specifies the source for the right input of Multiplier 3.

Code Point	Source
0	Z2
1	Z3
2	Y1
3	R3

Field 62. The M4 Left Source field specifies the source for the left input of Multiplier 4.

Code Point	Source
0	X1
1	X2
2	Y2
3	Y1

Field 63. The M4 Right Source field specifies the source for the right input of Multiplier 4.

Code Point	Source
0	Z1
1	Z4
2	Y2
3	R4

The data from this appendix are compiled in foldout form in Table B1.

Appendix C
KRONOS TIME-SHARING SYSTEM

Signing On

1. Put the terminal in half duplex mode.

2. Select the desired speed: 10 characters per second (CPS) or 30 CPS.

3. Dial the computer: (301) 340-2400 for 10 CPS; (301) 340-2440 for 30 CPS.

4. When the computer responds with a high-pitched beep, perform the appropriate coupling action (e.g., put the phone receiver in the acoustic coupler).

5. The computer should print

```
            yy/mm/dd.     hh.mm.ss.
EASTERN CLUSTER CTR KRONOS 2.0.8 SYS B
USER NUMBER:
```

where yy/mm/dd is the date and hh.mm.ss is the time of day. Type your user number, a comma, and your password followed by a carriage return (CR).

6. The computer should then print

```
TERMINAL:        number
RECOVER/SYSTEM:
```

which indicates that the sign-on procedure is complete, and you may proceed with your work.

CREATING A NEW FILE
(FOR EXAMPLE, AN AMIL OR ANIMIL PROGRAM)

1. Enter

```
NEW, filename CR
```

where filename is the name of the file being created. The computer should respond

```
READY.
```

2. Simultaneously press the CONTROL and B keys. The computer should respond with

```
TEXT MODE — EXIT BY CTRL C
```

3. Enter the information the file is to contain, one line at a time.

4. Simultaneously press the CONTROL and C keys. The computer should respond with

```
EXIT TEXT MODE.
```

5. Enter

PACK CR

to put the file in the proper internal format. The computer should respond with

```
READY.
```

6. To save this file for later use enter

SAVE CR

to which the computer should respond

```
READY.
```

SIGNING OFF

1. Enter

BYE CR.

2. The computer should print

```
account number    LOG OFF.     hh.mm.ss
     SS           sss.sss SEC  $   cost
```

and break the phone connection.

OTHER PROCEDURES

1. To use the programs described in this report, follow the directions in the separate sections.

2. To edit (change) a file (e.g., if an AMIL or ANIMIL program had errors), see the *KRONOS Text Editor Reference Manual*, Control Data Corporation Publication Number 59150700.

3. For additional information, see the *KRONOS Time Sharing User's Reference Manual*, Control Data Corporation Publication Number 59151300.